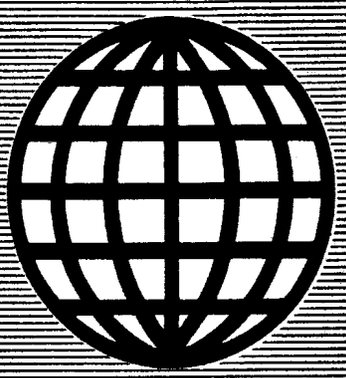


The COMPUTER JOURNAL

Programming - User Support
Applications

Issue Number 56 July/August 1992 US$3.95

TCJ - The Next Ten Years

Input Expansion for 8031

Connecting IDE Drives to 8-Bit Systems

Real Computing

8 Queens in Forth

Z-System Corner

Kaypro-84 Direct File Transfers

Analog Signal Generation

The Computer Corner

ISSN # 0748-9331

SAGE MICROSYSTEMS EAST

Selling & Supporting the Best in 8-Bit Software

(New Lower Prices on Many Items!)

- Automatic, Dynamic, Universal Z-Systems: Z3PLUS for CP/M-Plus computers, NZCOM for CP/M-2.2 computers (now only $49 each)
- XBIOS: the banked-BIOS Z-System for SB180 computers ($50)
- PCED — the closest thing to Z-System ARUNZ, and LSH under MS-DOS ($50)
- DSD: Dynamic Screen Debugger, the fabulous full-screen debugger and simulator ($50)
- ZSUS: Z-System Software Update Service, public-domain software distribution service (write for a flyer with full information)
- Plu*Perfect Systems
  - Backgrounder ii: CP/M-2.2 multitasker (now only $49)
  - ZSDOS/ZDDOS: date-stamping DOS ($75, $60 for ZRDOS owners, $10 for Programmer's Manual)
  - DosDisk: MS-DOS disk-format emulator, supports subdirectories and date stamps ($30 standard, $35 XBIOS BSX, $45 kit)
  - JetFind: super fast, extemely flexible regular-expression text file scanner (now only $25)
- ZMATE: macro text editor and customizable wordprocessor ($50)
- BDS C — complete pkg including special Z-System version (now only $60)
- Turbo Pascal — with new loose-leaf manual ($60)
- ZMAC — Al Hawley's Z-System macro assembler with linker and librarian ($50 with documentation on disk, $70 with printed manual)
- SLR Systems (The Ultimate Assembly Language Tools)
  - Z80 assemblers using Zilog (Z80ASM), Hitachi (SLR180), or Intel (SLRMAC) mnemonics, and general-purpose linker SLRNK
  - TPA-based ($50 *each* tool) or virtual-memory ($160 *each* tool)
- NightOwl (advanced telecommunications, CP/M and MS-DOS versions)
  - MEX-Plus: automated modem operation with scripts ($60)
  - MEX-Pack: remote operation, terminal emulation ($100)

Next-day shipping of most products with modem download and support available. Order by phone, mail, or modem. Shipping and handling: $3 USA, $4 Canada per order; based on actual cost elsewhere. Check, VISA, or MasterCard. Specify exact disk formats acceptable.

Sage Microsystems East
1435 Centre St., Newton Centre, MA 02159-2469
Voice: 617-965-3552 (9:00am - 11:30pm)
Modem: 617-965-7259 (pw=DDT) (MABOS on PC-Pursuit)

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler

Technical Consultant
Chris McEwen

Contributing Editors
William P. Woodall
Matt Mercaldo
Tim McDonough
Frank Sergeant
Clem Pepper
Richard Rodman
Jay Sage

The Computer Journal is published six times a year and mailed from *The Computer Journal*, P. O. Box 535, Lincoln, CA 95648, (916) 645-1670.

Opinions expressed in *The Computer Journal* are those of the respective authors and do not necessarily reflect those of the editorial staff or publisher.

Entire contents copyright © 1992 by *The Computer Journal* and respective authors. All rights reserved. Reproduction in any form prohibited without express written permission of the publisher.

Subscription rates within the US: $18 one year (6 issues), $32 two years (12 issues). Foreign (surface rate): $24 one year, $44 two years. Foreign (airmail): $38 one year, $72 two years. All funds must be in U.S. dollars drawn on a U.S. bank.

Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 535, Lincoln,CA 95648.

Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used trademarks are acknowledged, and we apologize for any we have overlooked.

Apple II, II+, IIc, IIe, Lisa, Macintosh, ProDos, Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. DateStamper, BackGrounder II, Dos Disk; Plu*Perfect Systems. Clipper, Nantucket; Nantucket, Inc. dBase, dBASE II, dBASE III, dBASE III Plus, dBASE IV; Ashton-Tate, Inc. MBASIC, MS-DOS, Windows, Word; MicroSoft. WordStar; MicroPro International. IBM-PC, XT, and AT, PC-DOS; IBM Corporation. Z80, Z280; Zilog Corporation. Turbo Pascal, Turbo C, Paradox; Borland International. HD64180; Hitachi America, Ltd. SB180; Micromint, Inc.

Where these and other terms are used in The Computer Journal, they are acknowledged to be the property of the respective companies even if not specifically acknowledged in each occurrence.

TCJ The Computer Journal
Issue Number 56 July / August 1992

Editor's Desk 2

Reader-to-Reader 2

TCJ - The Next Ten Years 5
What is in store for TCJ readers.
By Bill Kibler.

Input Expansion for 8031 7
Using 74165 for embedded expanison.
By Tim McDonough.

The Z-System Corner 10
Jay gets ready for Zed-Fest.
By Jay Sage.

Connecting IDE Drives to 8-Bit Systems 11
Tilmann explains how.
By Tilmann Reh.

8 Queens in Forth 14
A classic study done in Forth.
By Frank Sergeant.

Real Computing 17
Linux, BSD 386, and MINIX news.
By Rick Rodman.

Kaypro-84 Direct File Transfers 19
How to connect without a null modem.
By Walter J. Rottenkolber.

Analog Signal Generation 22
D/A conversion on the cheap!
By Al Mitchell.

The Computer Corner 24
By Bill Kibler.

EDITOR'S COMMENTS

Here it is, the new and improved version of TCJ. It has taken considerably longer than I thought, but then doesn't anything with computers these days. Before I talk about the articles, a few comments on the new format.

I have included a small box at the top of each article to help readers understand what the article is about. It has three lines and typically the first will be the main idea. Our next ten years article is the feature article and as such the header states that fact.

Next item in the box is the level of computer skills needed or in the case of the feature article, it is for all readers. Lastly is a topics list that can help show some specific concepts that are covered in the article.

New in the format is MY comments at the start of each article. You can tell it's me, because they have my initials at the end (BDK). This way you get to read what I hope is some specific information that will get you off to a good start reading the article.

The articles are good as always. TCJ has many excellent writers and this issue is no exception. I have pressed some new people into covering some different topics. For comments from the editor, other than a quick review of what is inside this issue, you will need to check out my Computer Corner, where I have been making comments on many topics for over 8 years (hard to give up the old ways.)

Other than our feature, which is a must read for everyone, we have Jay Sage, Tim McDonough, Frank Sergent, Tilmann Reh, and our surprise new writers. So set back and check out the issue.

Reader to Reader

Dear TCJ:
I have gotten one of your free issues and have decided not to subscribe. I work for a small company and had hoped you would help me get up to speed with new microcontroller construction projects. So far it looks like everything is way over my head. I need beginner and refresher information. Sorry, but thanks for the freebee.
-John Clark, LA, CA.

Thanks for the letter John, and you will get this and one more issue. The reason is our change of ownership and some minor correction of course. You see John I have been well aware that we have not been covering the beginner for some time now. I feel that even experienced people start out on new projects as beginners as well. So when ever we start a new topic from now on, we will try and cover it from a beginners view point first. We can not however teach you everything about electronics, so I will assume you have had at least a few basic courses in electronics and know what end of a soldering iron to use. I have heard that most of the competiors have refused to print beginners articles, NOT HERE. I teach at a junior college and have learned how hard it is for most people to get started on new projects. That is another reason I am trying to do "Platform Independent" projects. The simpler the project the better chance you have of learning. Please write me again if you have topics or suggestions about projects that would make you a confirmed TCJ reader. -BDK

Dear TCJ;
I don't know how to write an article about my YASBEC, but I can write a letter, so here goes.....

You probably have already looked at the pictures I sent, so maybe this will help orient you. The case is Hewlett-Packard surplus I paid about $5 for. (I keep forgetting prepositions are things you're not supposed to end sentences with!) The case housed a power supply, I think. Outside dimensions are about 5 1/2" x 8 1/2" x 13 3/4". Front and back aluminum castings are connected with removeable rails. There are top, bottom and side removeable covers (detach with one screw). The back panel already had the ac power receptacles, switch and fuse, and it was a simple matter to cut the openings for the parallel and serial connectors. The 5 1/4" Mitsubishi 40 track drive attaches with screws to the bottom of the case, the two TEAC FD235 3 1/2" drives are stuck to the 5 1/4" drive with several thicknesses of double sided (removeable) foam. The YASBEC board is stuck to a thin plastic panel also with foam pads. The thin plastic sticks to the top of the 3 1/2" drive. Power is obtained from a 150 watt IBM Clone removed from its metal case. 1/8" deep by 1/8" wide vertical slots in the side rails near the rear of the cabinet allow the power supply board to fit neatly, allowing enough room at the sides for ribbon cable to the parallel and

continued on page 3

serial connectors. On the left side of the cabinet I mounted a Rodime 45 meg 3 1/2" hard drive, attached to the horizontal rails. I also placed a small (2 1/2") 12v fan behind the hard drive. The fan may be unecessary since the perforated covers allow much ventilation. The front of the panel is filled in with a bit of gold colored perforated aluminum cut to fit around the drives and holding the reset button and pilot LED.

Everything works well except the hard drive. That appears to be a BIOS problem and I am hoping the new banked bios will take care of that.

A bit of history..... I first learned about YASBEC in the TCJ editor's note in TCJ #50 and in Z-Letter. I watched with interest the TCJ and Z-letter references, then in the late summer talked with Paul Chidley on the phone. Didn't take the plunge until late fall then sent Paul my check. All the info on Genie and Zcentral was helpful in gathering the components. Ludo Hemelryck supplied the Z180, SCSI and some sockets. Once I had all the parts it was a breeze soldering them in. As far as I could tell there were no solder bridges or cold joints. My only error was to rotate the scsi socket 90 degrees. But the error was spotted and the socket modified with my dandy little Dremel Motortool. (Later I removed and replaced the socket.)

Then everything was hooked up. I turned on the power..... NO SMOKE. Hooked up the terminal, but nothing on the screen. It wasn't until Ludo told me to hit return a few times (and at last I perused the documentation)that I was able to get the screen display. About that time I received the BIOS disk from Cam Cotrill. At first I couldn't get zcnfg to work. It turned out my junkbox 5 1/2" drive was bad. With a decent drive it booted fine and I was able to configure the 3 1/2" drives. All this time the whole thing was only haywired together. I was moving the main board around when suddenly..... A small puff of smoke toward the front of the board!!!!! And all was gone.

Surely after all this my YASBEC couldn't be dead! But it was. I later learned I had been Albanyed. It seems the 5 volt supply on the board has a trace (hidden under a ram socket) that acts as a fuse if vcc is shorted. Paul Chidley told me about that. It seems a soul (who shall be nameless) from Albany NY had inadvertantly done the deed I did. I am also from Albany (Oregon). Not knowing about the New York incident I thought Paul was ribbing me. Each of the states of Georgia, California, Illinois, Indiana, Kentucky, Louisiana, Missouri, Ohio, Texas, or Wyoming also has an Albany. I wonder if anyone there has put together a YASBEC!!!!!

At any rate jumpering the burned trace put everything straight again.

Most Z software I have tried so far works. I'm using a Televideo 970 Terminal. The extended tcap for the 970 is almost right on screen graphics. I haven't yet found out why LSH doesn't work. Since I plan to go to Trenton in April I'm hoping to get some help.

To summarize my experience - 1) the YASBEC is a great little board 2)it went together easily 3) except for the hard disk everything works great 4)the folks in the 'Z' community are most kind and helpful. I trust this gives a hint of one YASBEC users experience

Yours Sincerely
A. A. Straumfjord, Albany, OR.

Thanks for the letter and pictures. Sorry I couldn't get the pictures in this issue. With all the changes, I am not ready to put pictures in yet, maybe next issue. Glad to see that some are doing and succeding with projects from our pages. Maybe if you learn some more about constructing, you can give us a beginners eye view of what type of problems you overcame to make it work. How about it? - BDK

Dear TCJ:
I am a regular reader of the Computer Journal, either by subscription or second-hand. I also see you on the CPMTECH Fido echo occasionally. I am pleased that you continue to support small developers and "classic" development environments and tools. To further those goals, I'd like to offer you an article or two that suggest yet another developers' strategy that combines low cost, familiar technology, flexibility, and support for Zsystem/Z80/Zx80 technology.

For several years I have worked with, bought and sold S-100 equipment. As an electrical engineer and programmer in the 70's and '80's, I often found these systems operated by my industrial clients as well as in the basements of my engineering colleagues. Today, I still find them in use among many technologists, both experienced and new; as well as in flea markets, basements, and Salvation Army stores. Many people still refer to the IMSAI, with its front panel and blinking lights, with affection.

As you know, there is a resurgence of interest in Z-80 based systems due to development in new operating systems (Z-system, et. al.) and new, faster, and expanded Z80 processors (Z180, Z280, Z80XXX I/O processors.) Yet it is relatively difficult to work with these processors and systems given the almost overwhelming influence of the MS-DOS world of cheap, fast, available 80X86-based computers: computers that are also hard to program at the "nuts and bolts" level. I believe I can offer a better alternative to your readers.

S-100 systems (or IEEE-696, an ANSI standard) can offer a very affordable platform for Z80 development. In fact, almost any processor can run on the S-100 bus (and many have!). While the speed of older designs was limited to 2-4MHZ, and the very oldest boards were limited in both speed and memory capacity: there are still many cards, built in the 80's, that have modest amount of memory, I/O capability and flexibility. More to the point, they were built on LSTTL chips that are readily available, quite rugged and - a big plus- well understood and documented. Given also that many S-100 boards have descriptive information printed on the board themselves, even boards purchased without documentation can be "deciphered" with a few evenings' work: an educational exercise in itself! Finally, cost are reasonable to free. A typical disk-based system sells - working and documented

- for $50 to $200. Undocumented and "orphaned" systems, about to be discarded, can often be had for the asking! I have already published two articles on the pre-IEEE S-100 bus in the Z-letter, which I hope will represents my writing ability and technological knowledge. I propose a short series of articles to introduce the possibilities of S-100 development systems as a cost-effective and easily-learned environment, culminating in a Z180 processor card design. The Z180 processors appear to be a "small stretch" from Z80's both in cost and capabilities, and more than adequate for most uses.

Regarding my own interests: As a reseller, of course, I am interested in encouraging sales of my various used cards (catalog enclosed). My experience suggests that my prices are not out-of-hand for the card replacement market; and I have priced many cards below those levels. The replacement market seems to be limited to Compupro/Cromemco owners who have strong commitments to maintaining systems: and even they are now "bailing out" at the few hundred dollar level. Meanwhile, I have many cards that are not purchased often: 16K static RAM cards, bank addressable and otherwise flexible; serial/parallel cards; and 8" floppy disk controllers. For developers and experimenters, these are still cheaper than the "single-chip solutions" and software that would replace them. An article on evaluating and adapting these cards, written in a methodological fashion for general use, would help ALL those people who are trying to get a system up without documentation. Finally, your vendors can provide Z-system software, and I can provide CP/M 2.2 for about $25.

In summary, I propose a series of articles as follows:

"S-100 Systems: The Final Frontier" would introduce S-100 and IEEE-696 systems as stable standards for small processor development. They are widely available, terribly economic, and suffer only from "old technology"(actually an advantage) and lack of documentation (a "learning situation" with the right tools and information).

"Buying and Fixing Your S-100 system" would describe the history of several S-100 manufacturers; suggest the best kinds of boards and systems to buy, and offer tips, techniques, and general manufacturing practices that can aid in debugging and testing your system without elaborate tools and technology. This can easily be two articles! Given that there is two months between them, however, I would like to encourage both purchase and experimentation early in the series.

"A Z180 for the S-100" would describe a straightforward design of a Z180 processor card with on-board static memory and serial ports, which would drive either the S-100 bus (briefly, two 8-bit data paths, and 16 bits of address) or the IEEE-696 bus (8 or 16 bit data paths, 24 bits of address).

So what do you think? Contact me promptly: obviously this is a significant commitment on my part, and I need to plan for it as soon as possible. Please send appropriate information on writers' guidelines and your general comments. Thanks for your efforts and interests!

Herb Johnson, Trenton, NJ.

Well herb that was some letter and I couldn't have said it better. I have been a long time user of the S-100 systems and in fact still have several running units. That catalog was missing and hopefully you will send me one soon.

What the S-100 systems reminds me of, is when people were just starting to become electronic technicians, a normal approach was tearing apart and repairing old TVs. Well today, the modern computers have nothing that can be fixed or tinkered with in them. I have been thinking of doing some beginners and learning articles on using the S-100 products as the basis of building the readers skills. I will be calling you soon and dropping in the mail the writers guides as well. For our readers information then, you can expect to see some of Herbs' work in up coming issues. Please check over the Back Issues section of the magazine and you will find we have done many S-100 articles over the years. In fact I though we had already done a Z180 S-100 construction project as well.

Having written several of the past S-100 articles, and cut my teeth on them, it will be like old times to run your articles. And by the way, do you happen to have CP/M 68K for a Compupro CPU68K? As far as documents, I would like to point out to our readers that I and many of our readers have hung on to our old manuals. So if you are having troubles getting one of those S-100 systems running drop me a card and if I can't send you copies of what you need, I will at least put your request in the Reader-to-Reader section. If we get enough request for "orphan information", I will set aside a special section and maybe ask Herb to be our Doctor S-100. Now that request for literature can be for systems other than S-100 as we have readers who still use all types of non-PC products. BDK

Reader-to-Reader

To send letters to TCJ, mail them to:
The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

The editorial policy is to try and print your entire letter "as is." TCJ however does reserve the right to reject and or modify (by omitting) portions of letters deemed unfit for pulbication.

Major letter and minor articles are accepted on floppy disk and will aid in getting your letter published "as is." TCJ does not return disks and material, unless suitable and appropriate return mailers are provided.

Floppy disk and word processor formats supported are all the generally accepted programs (I.E. Wordstar and WordPerfect) as well as any format acceptable to PageMaker 4.0. CP/M formats are supported through use of a CP/M to PCDOS disk format converter. Currently no 8 inch disk formats supported (later maybe.) GENIE mail checked on regular basis (use B.KIBLER.) Other bulletin boards and services will be added as time permits.

The Computer Journal - The next TEN years

Feature Article

All Readers

Future of *TCJ*

By Bill Kibler Editor/Publisher

When issue 60 rolls off the press at the end of this year, The computer Journal will mark ten full years of "how to" articles. It seems as if it was a short time ago that I wrote that first TCJ article. I have in fact been writing for TCJ practically the whole time and plan on continuing to do so. What the next ten years will bring in computing I have no clear ideas. What TCJ will bring to you I want to talk about.

THE BEGINNING

When Art Carlson started The Computer Journal, it was mainly an outlet for do it yourself hardware projects. Art had many projects he wanted to do, and felt that other would want to know the results of his endeavor. My first article for Art was on upgrading 16K memory cards to 64 and 256K. Another article was on adding a Centronics interface to a Z80 system. Many smaller hardware projects followed as well as the software to support those changes.

I would say those first few years, saw equal amounts of hardware and software features. We also added industry and machine specific columns to help keep our readers abreast of changes that might effect them. The main focus were hardware "how to" articles. We have changed some over the years and I am not sure for the better.

THE NOW

The industry as a whole has moved away from "doing hardware." TCJ has been influenced by this change as well. The influence of the PC with easily assembled hardware and low price have made hardware hacking a thing of the past. The big caveat is that hardware is and always will be needed, and some of us love to tinker. As a HAM radio operator I got started by building and modifying transmitters and receivers. In college I studied solar system design and intended on using computers to control solar house systems. At that time microcomputers were just becoming available and my love of tinkering put the two together.

The modern PC clone is no longer a box to tinker with. The operating system and software needed is so complex the supporting documentation stands over three feet tall. It is not the machine for people to learn on and have fun with. When it comes to doing writing and design the PC works just fine. I think it should stay that way myself. So what has happened to TCJ over the years has been getting caught up in the PC clone hardware mess without intending to do so.

TCJ has had many hardware articles, in fact Chris McEwen got Frank Sargent to give us his award winning RTX system. That was a non-PC clone hardware and software project that given the time I will build myself. I can build that project because I happen to have an RTX contest board. What if you don't have one however and what about the RTX chip itself?

THE NEXT TEN

Over the next few issues and on into the future, TCJ will be making a few minor changes. I plan on taking the magazine back to our "ROOTS". With that I mean more articles on hardware. More how to do it articles for people of all experience levels. One problem with that approach is usually the platform of choice. My choice is NONE! The articles will be "platform independent". That means they must be projects based on serial or parallel ports, if not standalone.

My requirements do not stop there. No PALS or specific vendor devices allowed. These projects must be possible to build in any garage anywhere in the world that has access to standard computer parts. For the software side, I will stress FORTH as the language. Forth is the only true platform independent language around. The articles will however have other languages, especially assembly when needed. The article will help guide and show the reader how to do it the most efficient way possible. Pseudo code or flow chart like coding will be used so that each reader will be able to convert the given code to a language of their choice. TCJ will not be another "C" only magazine.

Market Standing

The Computer Journals standing in the market place will be somewhere between Circuit Cellars INK, Elektor, and Midnight Engineering. In reviewing those issues, each has something for our readers. They also have points of view or positions that make them less effective for our readers. Take Circuit Cellars INK for example. The magazine's focus is on selling Micro Mint products which are almost always based on Intel products. Currently they feature wiring the house with Micro Mint systems if you want to be "in with the electronic cottage."

Readers of Midnight Engineering appear to be hopeful business people. The magazine header is very clear about their support of people wanting to make a business out of their hobby. There are some very good articles on hardware and software, but business articles are their main stay. I am not sure that our readers are so business directed that they want business articles in place of more how to hardware articles. I feel that many of our readers are using computer products to sell and service non-computer based products. They know their business and how to package it, what they

need is how to save money using this system or that process.

Elektor magazine is new and very close to my idea of what level of support we should be providing. The problem with Elektor is they also support radios, antennas, audio amps, and computers. That broad approach is fine unless your interests are only learning how to use stepper motors. Why pay for a magazine that only has one article an issue of interest to you.

What my hope for TCJ is that we can provide the reader with a full issue of important and interesting articles. To that end I propose trying to have a 30% ratio. I want a third of each issue to support old or recycled systems. Many of our foreign readers (15% of our subscribers) don't have access to the latest systems. Currently no other magazine supports Z80, S100, or ZCPR users. We will continue to provide regular and special articles for those users.

The next third will be on hardware and software projects (more hard than soft I hope) that are platform independent. Serial and parallel projects that will lead the reader through the steps of building and having a useful device in the end. Like most magazines we will try to provide either bare boards or complete kits for those without access to the materials used. I have had several requests for stepper motor how to. We have talked about steppers in the past, but too much space was put on software and not enough on the hardware. The person who pointed this out had blown up several stepper motors before giving up completely.

The last third will be on reviewing products and services important to our readers. I have often complained that Forth Interest Group never reviews vendors of Forth. If your starting to look for a Forth that will help you talk to remote door openers, whose is the best, or do I roll my own. Now reviewing has its' problems, as you can appear to represent one or the others product. Our method of reviewing will be two fold. First I will let the vendor explain the why and wherefore of their product. This will include some examples from their manual (usually the tutorial section). Then I will find someone not related to the vendor and ask them to try the tutorials and produce some real results. That will take several articles to cover, but by the end all the good and bad points should be clearly visible. The object is for TCJ not to give an opinion on the product but to give you enough information to make up your own mind. What are we going to review, I got two items to start with, embedded controllers development packages and CPU trainers.

One main issue that TCJ will continue to press is keeping things affordable. Our aim will be to make The Computer Journal the most cost effective magazine you receive. We will do that by putting more articles that you can use and build. Those projects will also be as inexpensive as possible. Recycling of systems will be one strong point of hardware projects.

The Next Step

At this point I need to move on to being more of an editor than writer. The next issue of TCJ is starting to come together and as such I must put off more comments on TCJ's future. As readers and supporters of TCJ, you can continue to express your thoughts about where we need to go. I always look forward to reading and publishing your comments. These minor changes and TCJ's next ten years are really the choices of you, the reader. Without your input TCJ will be unable to continue it's growth. We have doubled in numbers from Chris's work and maybe we can do that again with this improved focus. Let me know what you think.

As always TCJ can be found on GENIE as "B.Kibler" and a answering machine is standing by 24 hours a day for your subscription or comments. The new address for your correspondence is P. O. Box 535, Lincoln, CA 95648. That phone number is (800) 424-8825, or (916) 645-1670.

UNIVERSAL MICROPROCESSOR SIMULATOR/DEBUGGER 2.0

- Simulates the Z80, 8051, 8085, 6811, 6809, 6805, 6801, 6800, 6303, 6502 & 65C02.
- Cross Assembler and Disassembler.
- Accepts Binary, Motorola, and Intel Hex codes.
- **Windowed source-level Debugger.**
- Handles exceptions simulation.
- Includes batch file capability.
- Utilizes your PC's I/O for MPU simulation.
- **$90** each set.

THE ROMY-8 EPROM EMULATOR

- Works with CPU simulator (Exclusive Feature).
- Lets you change and test code in seconds.
- Monitors address bus.
- Patch code with assembler.
- Emulates 2716-27256 EPROMs.
- Loads 32K of code in 20 seconds (PC/AT).
- 90 day warranty.
- Saves you money - only **$155** (complete with one set of CPU simulator).

J&M Software Hardware Design, Inc.
83 Seaman Road, West Orange, NJ 07052
TEL: 201-325-1892 FAX: 201-736-4567

INPUT EXPANSION FOR 8031

By Tim McDonough

Embedded Systems

Beginners and up

8031 Construction

Tim's article this time is a simple hardware method to get more I/O for your system. It is based on his 8031 system he sells, but the technique is deffinitely PLATFORM INDEPENDENT. As you will see he uses a chip I have seen in the books many times but never thought to use it in this way. Great idea Tim, BDK.

Digital inputs in a microcontroller system are like disk space on your PC or vacation days at the office --you can never have too many of them and, more often than not, things would be better if there were just a couple more...

Are your projects like many of mine? Do you often wish that all 40 pins on an 8031 were inputs? Having every pin as an input port may be impossible but there is a method you can adopt that will let you stretch 4 of the existing I/O lines to nearly as many digital inputs as your system will ever need.

The traditional method of adding digital I/O to many microcontroller systems is the addition of an 8255 Programmable Interface Adapter (PIA) into the memory map of the system. The 8255 is addressed as RAM and provides 3 bi-directional ports of 8-bits each for a total of 24 additional I/O lines in your system. It is a very reliable component and is well worth considering for many applications.

There is some downside to the 8255 style of I/O expansion, particularly if what you need are more input lines -- it is a memory mapped device. If your system requires a large amount of RAM several ICs or a Programmable Logic Device (PAL) may be required to decode the PIA into the smallest amount of memory space possible, thus preserving the maximum amount of space for system RAM. In other applications the circuit board space and the cost of the extra decoding logic may be unacceptable.

My digital input expansion example requires 4 of the 8031's I/O lines to provide any number of additional digital inputs. A little software overhead is required but the advantage is that whether you need 8, 16, 24, 32 or even more inputs, this method only requires 4 of the MCU I/O lines leaving as many as 10 lines free for outputs or special purpose inputs if you don't require a serial communications port.

The key to the design is the 74HCT165N Shift Register. This IC is a parallel-in, serial-out register that easily converts the digital inputs in a system to a serial bit stream that can be collected by the 8031. Inputs and outputs on the '165 allow them to be cascaded to provide any number of inputs subject to the fan-in/fan-out limitations of the LOAD, CLOCK, and ENABLE lines that are bused together.

The project is based on my Control-R I Single Board Computer (SBC). It provides the core components used for many 8031 designs. Major components include an 8031 MCU, address latch, EPROM socket, and MAX232 to provide an RS232C compatible serial port. Serial communications routines for the 8031 have been presented in other TCJ articles and have been included in the source code for the system I'm presenting here.

The 8031 has a total of 16 digital I/O lines available for general purpose use. Most of my applications use the serial port which requires a minimum of two of the I/O lines on Port 3 (more if hardware flow control is required.) If the system includes external RAM or other memory mapped devices, two additional lines on Port 3 must be used for the *RD and *WRITE signals. The bottom line is many applications require a lot of digital inputs. Configuration switches, operator controls, sensors, etc. can rapidly consume the remaining 12 I/O lines before your project gets too far off the ground.

Figure 1 shows a schematic diagram of a pair of 74HCT165N shift registers that I've attached to the basic Control-R I SBC. These two registers provide the demonstration system with 16 bits of digital input. Additional 74HCT165Ns may be cascaded to add more input lines as desired. Note that each shift register input has a 10K pullup resistor. These pullups are mandatory to force each input into a stable condition (logic "1"). Failure to include them in your design may have disastrous effects on your system.

In many systems it may not be strictly necessary to use the ENABLE signal. If the shift registers were the only circuitry attached to the lines used for DATA, CLOCK, and LOAD it could be tied to ground, permanently enabling the registers. I prefer to control it from the 8031 so that I can use the port pins assigned to DATA, CLOCK, and LOAD to other functions if needed.

Programming the shift registers is quite straightforward. Listing 1 shows the complete source for the application. The essential steps in moving data to the microcontroller's memory are as follows.

1) The ENABLE line of the shift registers is brought low.

2) The LOAD line is briefly pulsed low to latch the data present on the parallel input lines into the register

3) For each input line present, a positive going clock pulse is applied to the CLOCK line and the state (high or low) of the DATA line is read and processed by the 8031 using one of the bit test instructions.

4) The ENABLE line is brought high, disabling further control of the shift register.

The SHREG and READ8 subroutines are the key pieces of code for reading the shift register data. SHREG enables the shift registers and loads data into the shift registers. Next, for each shift register in the system, the READ8 subroutine is called. READ8 returns the eight bits from a register packed into a single byte that is returned in the Accumulator register. After each call to READ8, the SHREG routine stores the returned data in memory for later use.

The READ8 subroutine is easy to understand once you walk through it a couple of times. It starts by loading the number of bits to be read from a single shift register into R0 (Register 0) which is used as a counter. Next, the clock line attached to the shift registers is pulsed high to shift one bit of data onto the DATA input to the 8031. The next step is to clear the 8031's "carry" flag in preparation for processing the bit.

The bit test is done with the 8031's JNB instruction. The DATA line is tested. If it is "low" or logic zero, program execution jumps to the code at label "READ82". If it was "high" or a logic one, the SETB (Set Bit) instruction sets the value of the carry flag to a logic one. At this point my program has set the carry flag equal to the value of the DATA line.

The next step is to move the data that's been stored in the carry flag to a temporary variable that is used to hold the other bit values as well. First the temporary data is copied into the Accumulator. Then a RLC (Rotate Left with Carry) instruction copies the bit that was stored in the carry flag into the least significant bit of the accumulator. All other bits in the accumulator are shifted one to the "left" as well. Finally, the current value is placed back in the temporary storage location R1 (Register 1).

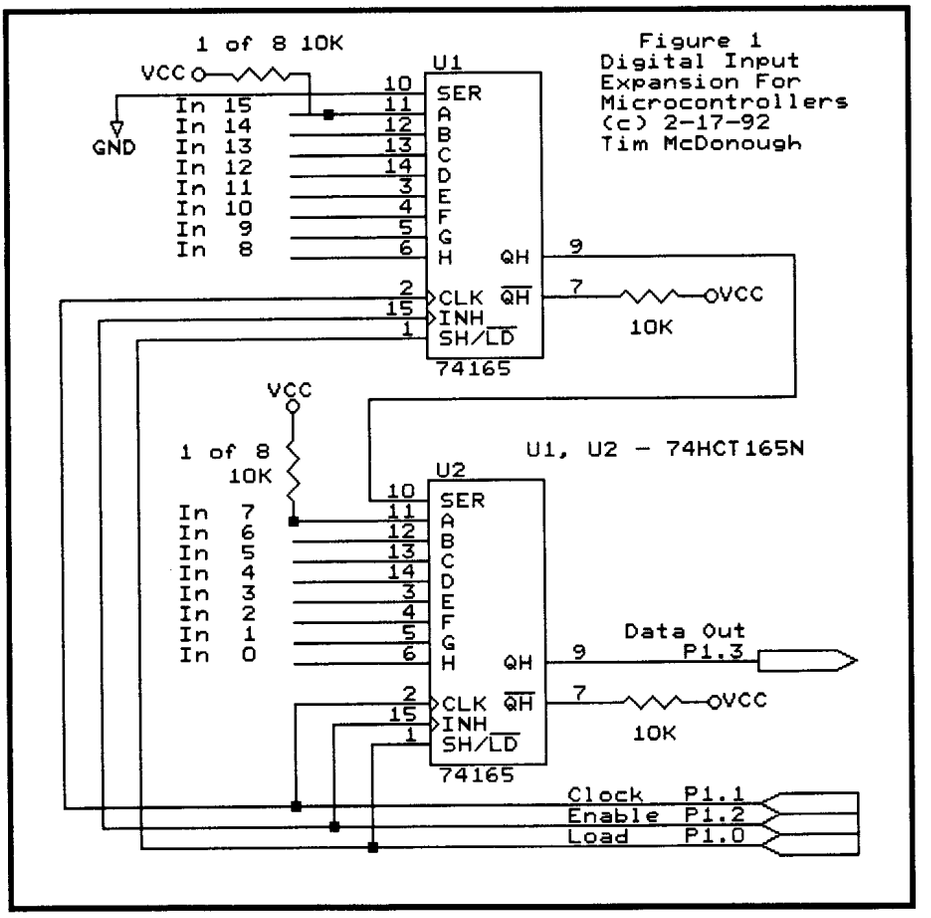
Once a bit has been inspected and stored in the temporary variable R0 is decremented and tested. If it's not equal to zero, meaning all eight bits haven't been read yet, program execution jumps back to label READ81 for the next bit. If it is zero, all eight bits have been read. The temporary value is copied into the accumulator and the subroutine returns.

The remainder of the program in Listing 2 should be old hat to experienced 8031 programmers or those of you that have read previous "Embedded Systems for the Tenderfoot" articles. Each time the command string "@A" is received at the serial port, the 16 digital inputs are read and stored in two one-byte memory locations. The data collected is then converted into printable ASCII form and transmitted back to the host computer as 4 hexadecimal digits.

While testing and debugging the hardware and software it can get very annoying to constantly have to run a terminal program and type "@A" while trying to measure what's going on in the circuit. To simplify my life (isn't that what computers are all about anyway?) I wrote a short program in Microsoft QuickBASIC Version 4.5 that polls the Control-R I for data once a second. The source code for the test program is shown in Listing 1.

I hope this latest installment in "Embedded Systems for the Tenderfoot" has peaked your interest. If you have questions or would like to see a specific topic covered in a future article please drop a note to TCJ and we'll see what we can do. I won't promise to do your next design project for you but I'll try to cover things that may be of general interest to developers just getting started.

For those of you who hate typing or who don't have access to Microsoft QuickBASIC for the test program, I'll make your life easy. If you live in the US or Canada, send a $5.00 (US) check payable to "Cottage Resources Corporation" to the address below and I'll send



you a 5.25" 360K MSDOS disk with the SHIFT.ASM source code, the compiled QuickBASIC test program, and some other assembly language odds and ends thrown in for good measure. Please indicate the check is for the SHIFT.ASM disk and include your return address. You cannot order this disk from the Sandy, Utah office and we will not accept credit card orders for it.

Tim McDonough is a network engineer turned electronics entrepreneur. He may be reached at the Midwest offices of: Cottage Resources Corporation Suite 3-672, 1405 Stevenson Drive Springfield, IL 62703 (217) 529-7679

'Listing 1:

```
' PROGRAM:        BITGET.BAS
' REVISION:  1.0
' DATE:     February 17, 1992
" AUTHOR: Tim McDonough
'          Cottage Resources Corporation
' This program collects data from an 8031 system with 16 digital inputs.
' "@A" requests the inputs be sampled. The data is returned as 4 hexadecimal
' digits.
DEFINT A-Z
OPEN "COM2:1200,N,8,1,BIN,CD0,CS0,DS0,RS" FOR RANDOM AS 1
DO WHILE INKEY$ = ""
          PRINT #1, "@A"
          INPUT #1, A$
          PRINT "Sample: "; x, A$
          x = x + 1
LOOP
END
```

Listing 2:

```
 .command +H60
 ; PROGRAM:        SHIFT.ASM
 ; REVISION:  1.1
 ; DATE:     February 17, 1992
 ; AUTHOR:        Tim McDonough
 ;          Cottage Resources Corporation
 ; This program demonstrates the use of parallel-in, serial-out shift
 ; registers to expand the input ports of an 8031 microcontroller. This
 ; example uses two type 74HCT165 intergrated circuits. By adding
 ; additional shift registers and increasing the buffers and clocking any
 ; number of inputs may be added while still requiring only 4 MCU lines.
 ; ; In this example program a host computer sends a command to the 8031
 ; that causes 16 external inputs to be read and stored in internal RAM.
 ; The two data bytes are then formatted as printable hex numbers and
 ; transmitted back to the host computer.
          .ORG      D'00      ;Assemble to start at location 0 Hex
 ; Equates
          .EQU      LOAD,P1.0 ;*LOAD signal to 74HCT165
          .EQU      CLOCK,P1.1          ;CLOCK signal for the 74HCT165
          .EQU      ENABLE,P1.2         ;ENABLE line for 74HCT165
          .EQU      DATA,P1.3 ;Serial data from 74HCT165
          .EQU      BUF1,H'7E ;1st data buffer
          .EQU      BUF2,H'7F ;2nd data buffer
 ; Initialization
 ; This section sets up the 8031's serial communications port for 8-bit
 ; operation at a specified baud rate. It assumes an 11.0592MHz crystal
 ; is used in the system clock circuit.
 ; The shift registers are read once and the data ingnored to clear out
 ; any stray bits from power up, etc.
          MOV       SCON,#H'52          ;Mode 1, 8 bit word
          SETB      TR1       ;Enable Timer #1
          MOV       TMOD,#H'20          ;8-bit auto reload, free running timer
          MOV       TH1,#H'E8 ;1200 baud operation
          ACALL     SHREG     ;Collect data and throw it away to clear
          ;out any junk
          SJMP      MAIN
 ; Main Program
 ; "@A" is the command string that must be sent to the system to request
 ; that the shift registers be read and the data transfered to the serial
 ; port.
 MAIN:    ACALL     RECV      ;Wait for a character
          CJNE      A,#'@',MAIN         ;Check for the '@' symbol
          ACALL     RECV      ;Wait for another character
          CJNE      A,#'A',MAIN         ;Check to see if it's 'A'
          ACALL     SHREG     ;Valid command, go read the inputs
          ;and store data in BUF1 & BUF2
          MOV       A,BUF1    ;Get data from buffer 1
          ACALL     SEND16    ;Transmit the value back to the host
          MOV       A,BUF2    ;Get data from buffer 2
          ACALL     SEND16    ;Transmit the value back to the host
          ACALL     CRLF      ;Send a carriage return & line feed
          SJMP      MAIN      ;Go do it again
 ; RECV Subroutine
 ; The receive subroutine waits for a character to arrive at the serial port
 ; and returns it in the A register when it arrives. This routine will not
 ; return until a character has been received.
 ; Effected registers: RI
 RECV:    JNB       RI,*      ;Wait for a character
          CLR       RI        ;Clear the receive flag
          MOV       A,SBUF    ;Move the character into A
          RET
 ; SEND16
 ; Transmits the byte in the A register to the host formatted as a two
 ; digit hexadecimal character.
 SEND16: MOV        R0,A      ;Store the byte in R0
          SWAP      A         ;Swap the high and low nibbles
          ANL       A,#H'0F   ;Zero the low nibble of A
          ACALL     HEX2A     ;Convert it to ASCII
          ACALL     SEND      ;Transmit it
          MOV       A,R0      ;Load the byte from R0
          ANL       A,#H'0F   ;Zero the low nibble
          ACALL     HEX2A     ;Convert it to ASCII
          ACALL     SEND      ;Transmit it
          RET
 ; HEX2A
 ; Converts a byte to an pair of hex format ASCII characters
 HEX2A:  CLR        C         ;Clear the carry flag
          SUBB      A,#H'0A   ;Test for 0-9, A-F
          JNC       HEX2A1    ;If no carry then it was A-F
          ADD       A,#H'3A   ;Add offset for 0-9 range
          RET
 HEX2A1: ADD        A,#H'41   ;Add offset for A-F range
          RET
 ; SEND Subroutine
 ; The send subroutine transmits the character or byte passed in the A
 ; register out the serial port once the 8031's internal UART is ready
 ; to accept a character. This routine will not return until the character
 ; has been transmitted.
 ; Effected registers: TI
 SEND:    JNB       TI,*      ;Wait until the transmitter is ready
          CLR       TI        ;Clear the transmit flag
          MOV       SBUF,A    ;Copy the character from A into SBUF
          RET
 ; CRLF Subroutine
 ; Transmits a carriage return/linefeed sequence out the serial port
 CRLF:    PUSH      ACC       ;Save ACC just in case
          MOV       A,#D'13   ;Carriage return
          ACALL     SEND      ;
          MOV       A,#D'10   ;Linefeed
          ACALL     SEND      ;
          POP       ACC       ;Restore the ACC
          RET
 ; SHREG Subroutine
 ; Gathers data from the two shift registers and stores it in buffer
 ; memory. Any number of shift registers may be added by allocating
 ; sufficient storage buffers and calling READ8 the correct number of
 ; times.
 SHREG:   CLR       ENABLE    ;Enable the shift registers
          CLR       CLOCK     ;Ensure the CLOCK line is low
          CLR       LOAD      ;Bring LOAD line low to parallel load
          SETB      LOAD      ;Bring LOAD line high
          ACALL     READ8     ;Read first 8 bits and
          MOV       BUF1,A    ;save the results in BUF1
          ACALL     READ8     ;Read the next 8 bits and
          MOV       BUF2,A    ;save the results in BUF2
          SETB      ENABLE    ;Disable the shift registers
          RET       ;Return to calling program
 ; READ8 subroutine
 ; Gets data, 8 bits at a time, from external shift registers and returns
 ; one byte in the accumulator. Must be called from SHREG for proper opertaion
 READ8:   MOV       R0,#D'08  ;Initialize the bit count to 8
 READ81: SETB       CLOCK     ;Raise CLOCK line
          CLR       CLOCK     ;Lower CLOCK line
          CLR       C         ;Clear the carry flag
          JNB       DATA,READ82         ;If DATA = 0, goto READ82
          SETB      C         ;DATA = 1, set the carry flag
 READ82: MOV        A,R1      ;Copy current data into ACC
          RLC       A         ;Rotate the carry bit into ACC
          MOV       R1,A      ;Store the new value into the buffer
          DJNZ      R0,READ81           ;Get another bit if R0 <> 0
          MOV       A,R1      ;Copy data into ACC
          RET
          .END
```

Regular Feature

ZCPR Support

Some General Thoughts

The Z-System Corner

By Jay Sage

I was able to talk to Jay just before he left for Europe. I had just finished "The Next Ten Years" article and faxed it to him. I think it was too much too fast for him. Sorry Jay, but I have been thinking about these changes for some time now. Jay pulled through and managed to give us some of his wonderful insights into a DOS program (from a lover of 8-bit systems no less - what is the world coming to,) BDK.

By now you are aware that TCJ has a new editor and publisher. During the period while TCJ was in transition, no one was enforcing submission deadlines. With no deadline, it was natural for other pressing tasks to come to the fore, and as a result, two days before I depart for my annual vacation in Germany I have no column ready!

I have a long list of things I want to talk about in TCJ, but there is no time to work out one of my usual complex, technical articles. So this is going to be a short, chatty column. At first I thought about just skipping an issue, but I have a strong commitment to tradition, and in the end I could not face the pain of breaking my uninterrupted series of columns going back to issue 25, more than five years ago.

Zed-Fest Europe 1992

Last year during my vacation I had the pleasure of participating in an exciting meeting that was organized by Helmut Jungkunz and Uwe Herczeg to bring together some of the most active 8-bit hobbyists from all over Germany (and elsewhere, too). Thus was born the first European Zed-Fest. We all had so much fun that we are doing it again this year.

In one of my TCJ columns last year I gave a sketchy description of last year's meeting. This year I am going over better prepared. I've even bought a roll of black-and-white film so I can take some pictures for publication. During the past month, whenever I exchanged Internet mail with Europeans, I have invited them to come to the Zed-Fest. As a result, I hope we will see some new people there this year. In my next column you can expect a thorough report.

4DOS: Z-System for DOS Computers

In my last column I already wrote enthusiastically about 4DOS, the ZCPR3 of the DOS world. If you can believe it, I have gotten another factor of 10 more enthusiastic since then. Just as Z-System enormously increased my productivity on 8-bit computers, so 4DOS has revolutionized my use of my DOS machines. Thanks to 4DOS facilities and the PMATE text editor, I expect to be busy at the office simulating a new circuit concept the whole time I am on vacation! This is even better than telecommuting. I'm going to let the machine do all the work for weeks on end while I relax in the mountains of the Black Forest.

The powerful flow-control facilities in 4DOS together with the extended environment variable functions will make it possible for my 486 computer to perform a complex set of SPICE circuit simulations to determine the parametric margins of a new circuit I have invented. A set of PMATE editing macros examines the SPICE output file and determines whether the circuit performed correctly or failed in some way. PMATE and 4DOS maintain a number of log files with summaries of the progress and results of the calculations. PMATE also writes information out to files that 4DOS can read to determine how to vary the circuit parameters. 4DOS can then conduct an efficient search for the parametric limits within which the circuit can operate. When I get back I expect to find a file with a concise summary of these results.

One new twist that I have to implement is power-failure recovery. We've been having lots of trouble with the power substation that supplies Lincoln Laboratory, and I have to expect at least one power failure while I am away. This means that I have to provide some way for 4DOS to save and restore its state. When the power comes back up, the computer has to restart the batch file and pick up where it left off.

I don't know if all this is going to come together before I leave, but I expect to give a report and describe the techniques I use in future issues of TCJ. Although this particular application is running on a DOS computer, the techniques apply under Z-System as well, and I use them regularly, albeit on a smaller scale.

Meeting the Author

Yesterday I stopped off at the offices of JP Software in nearby Arlington, Massachusetts, to meet with Tom Rawson, one of the two principal authors of 4DOS, along with Rex Conn who works out of a Washington office. My purpose in making this visit -- besides getting to meet someone whose work I so respect -- was to show them a few things that we do in Z-System that 4DOS does not yet do. The two main topics were error handling and the full-screen history shell capabilities of LSH. Tom was impressed, and I hope we will see some advances in this direction in a future release of 4DOS.

Everyone in the JP Software offices was unbelievably friendly, warm, and helpful. Tom agreed to meet with me prior to my departure for Europe even though a maintenance release of 4DOS was due out that very day and he had not quite put the finishing touches on it yet. Although we were talking about DOS, it felt like a meeting of Z-System enthusiasts. Not only does 4DOS -- the product -- have a Z-System feel, but so do the people involved in its development. I felt a strong sense of kinship. I really wish them well and hope that all you DOS users reading this column will purchase the commercial (or, if you prefer, registered shareware) version of 4DOS.

Have a nice summer everyone.

Connecting IDE Drives to 8-Bit Systems

By Tilmann Reh

Interfacing Systems

Intermedate Skills

8 - Bit Construction

I recently went to a IDE drive on my own system. Since then I have been wondering about their technical side. Although Tilmann is interested mostly in their 8-Bit data usage, the information presented here will help with any system. So Tilmann can I hook this to my NOVIX system? Read on and see for yourself. BDK

Most of us know about the features of a hard disk compared to floppy disk operation. You get much higher storage capacity while dropping access times down to a few milliseconds. Before installing a hard disk, I was used to copying all files for the current project to the RAM disk of my CPU280 and then copying all changed files back to floppy when I was finished working for the day. Although this is much better than working with floppy disks only, it is not comparable to using a hard disk. With a hard disk, you just work on your projects, which can now use files that would not fit on a floppy or RAM disk, and you don't have to copy files around the drives. In addition, the access times are almost as fast as those of a RAM disk. Last but not least, you are freed from changing floppies like a D.J., since all files are accessible without mechanical action on your part.

The Technology Decision

When thinking about connecting a hard disk to a given computer system, the main decision that must be made is which interface technology should be used. The old ST-412/506 interface is not up to date (I think it is impossible to buy new, small drives with that interface), and the hardware expense for this type of controller is great. Additionally, these controllers have some critical analog circuits which have to be adjusted very carefully.

The next technological step was the ESDI interface, which is quite similar to ST-412/506, except that data is transmitted in parallel and with higher(but still fixed) data rates. The ESDI controllers are more complex and the drives more expensive than the ST-506 components. So ESDI is not interesting for our use.

The SCSI interface is a universal peripheral interface that is often used for hard disk connection. Some machines (Apple Macintosh, NeXT) even support no other interface for this purpose. SCSI is a very powerful and good interface, and SCSI drives are well established and available at acceptable prices. For the host interface (that is the controller) there are some chips available that do the complete bus protocol work in hardware and software and deliver raw data for the host processor.

Another alternative is the IDE interface (also known as the AT Bus interface). This interface is used in almost all IBM clones these days. Like the SCSI drives, the IDE drives contain the complete hard disk controller on the drive (this is where the name comes from: IDE means "Integrated Drive Electronics"). But these drives are connected to the standard PC-AT bus system and are accessed just as with the normal PC-AT hard disk controller. So, in effect, you don't need a controller any more but just some interface electronics that simulates an AT bus. The IDE drives are slightly cheaper than similar SCSI drives, and since the interface is simpler, I chose this one for my project.

However, it must be said that this decision was made with only the hard disk connection in mind. If someone wants to connect more peripheral devices (i.e., scanners, CD ROM, etc.) or more than two hard disks (which is the IDE limit), the SCSI interface is preferable (one interface for all devices).

The Interface Circuit

The greatest problem when interfacing an IDE disk to an 8-bit computer system is the differing bus width. While the control registers of the IDE drive are still eight bits wide, the data transfer is done word-wise (16 bits each transfer). So we need an interface which maps the 16-bit data register to the 8-bit bus.

There are two basic approaches to doing that. The first (and easiest) one is to map the two halves of the 16-bit data into two different I/O addresses. The strobe for the IDE drive would have to be generated with the read signal for the lower half and with the write signal for the upper half. The control registers would then be accessed by always reading the LSB and writing the MSB. However, although only simple hardware is required, this method has a very great disadvantage: since two different I/O addresses must be accessed alternately, you cannot use string instructions (such as the Z80 INIR/OTIR) or a DMA controller for data transfer. Thus, the transfer rate would be relatively slow and the programming not very elegant.

The other approach is to map the 16-bit data onto two consecutive accesses to the same I/O address. This way, some circuit expense is necessary to switch the right data halves to the data bus and to

handle the 8-bit accesses to the control registers. However, with this slightly more complex hardware, we get great software advantages. With this technique, using string instructions or DMA is the normal way to transfer the data to or from the disk drive.

Of course, we need some circuitry to remember which half is to be processed next, in order to select the correct data path and generate the correct strobes for the drive. As we have to distinguish only two cases, one flip-flop is enough. Since I planned to use a GAL (generic array logic) for address decoding and bus interface anyhow, I used one of the GAL macrocells to make the flip-flop. The clock pulse for the flip-flop is generated each time any register is accessed, and the data is set to zero when other than the data register is selected. This way, accesses to any control register also reset the state flip-flop, thus ensuring proper conditions when the data transfer is started.

The rest of the interface circuit is self-explanatory: one half of the 16-bit data is always processed directly, while the other half is stored in a latch or register. For the control registers, the latch becomes transparent. The strobes and select signals for latches, buffers, and drive are generated with the GAL mentioned above.

Slack Space On Board

The IDE interface itself would easily fit twice (or even three or more times) on a standard EuroCard-sized PCB. To avoid wasting board space, I filled the free space with some useful circuits which would serve the CPU280 very especially well but also make sense with other systems. The first additional circuit is an active termination for the complete ECB Bus, which is absolutely necessary with bus clocks of 4 MHz or more and a backplane of some length. With still more space left, I added two control buttons for hardware reset and NMI (non-maskable interrupt) generation and four LEDs as a power-control monitor. Last but not least, I finished the design with a Centronics-type parallel printer interface. The decoding signals for this interface are generated using components that were already there for the IDE interface, so this involved almost no additional circuit expense. Of course, if someone needs only the IDE interface, they can just leave the rest of the board unused!

How To Get One

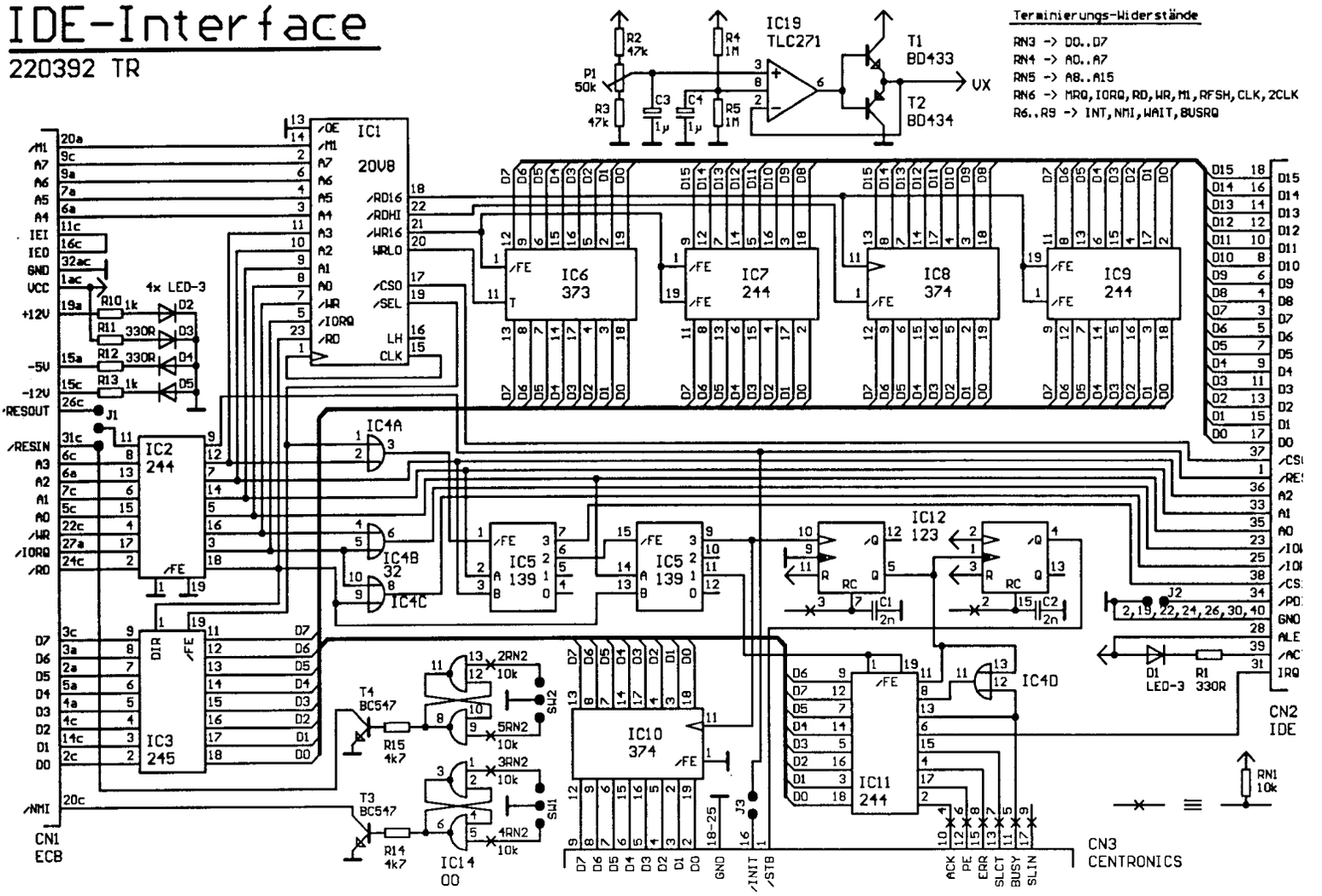
If you are interested in the interface, I think it will again be the best to contact Jay Sage for the availability of PCBs, programmed GALs, driver software, etc.

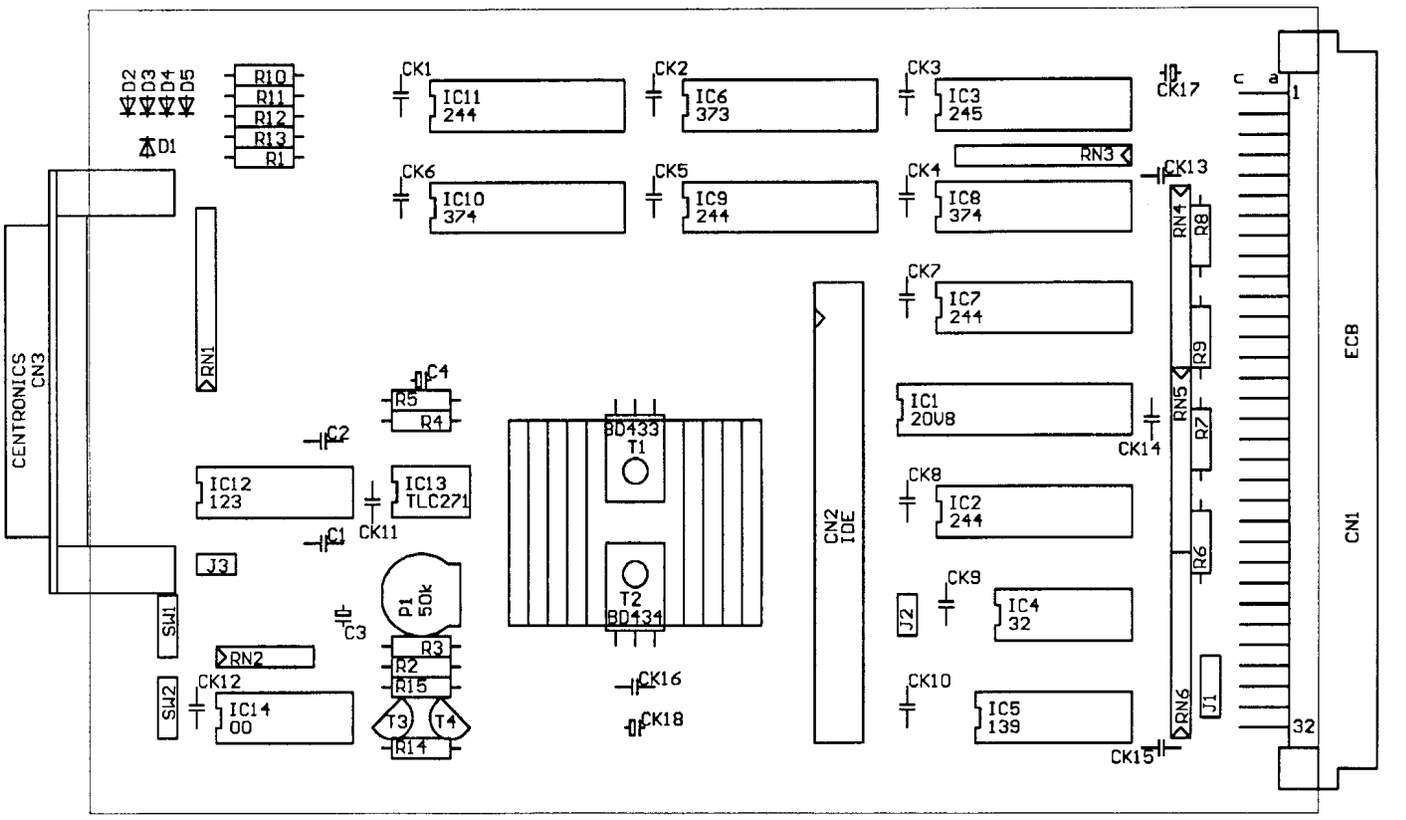
Tillmann Reh is an electronic engineer at the University of Siegen, Germany. He also owns a small company that develops custom solutions using embedded controllers or microcomputers. Tillmann has been active with CP/M since 1983 and developed a number of ECB-bus boards. He can be reached by regular mail at 'In der Grossenbach 46, W-5900 Siegen, Germany' or by E-mail (international/bitnet) at 'tilmann.reh@hrz.uni-siegen.dbp.de'.

```
TITLE          IDE/CENTRONICS INTERFACE GAL IC1
AUTHOR                TILMANN REH
COMPANY               REHDESIGN
DATE           22.03.1992

; Accesses to the hard disk are always with LH = high. So this signal
; has complementary meanings when reading resp. writing. To access the
; drive with the first data read, the LH flipflop has to be set before
; the real data transfer begins (one dummy-read of the data register).

CHIP IDE PALCE20V8

CK A7 A4 A5 IORQ A6 WR A0 A1 A2 A3 GND
OE M1 CLK LH CS0 RD16 SEL WRLO WR16 RDHI RD VCC

; Base address (BASE) to be changed only here! The lower nibble of the
; addresses are partly fixed by the hardware design.

STRING BASE '(A7 * /A6 * /A5 * /A4)'                ; Base Address 80h

STRING PARSEL '(BASE * /A3 * A2 * /A1 * A0)'   ; Centronics Adr. x5
STRING CS1ADR '(BASE * /A3 * A2 * A1)'         ; CS1        Adr. x6..x7
STRING DATADR '(BASE * A3 * /A2 * /A1 * /A0)'  ; CS0/Data   Adr. x8
STRING TFRADR '(BASE * A3 * (A2 + A1 + A0))'   ; CS0/Task   Adr. x9..xF
STRING IDEADR '(CS1ADR + DATADR + TFRADR)'          ; all IDE-Addresses

EQUATIONS

/CS0 = TFRADR                                       ; Task File Access
+ DATADR * LH                                       ; Data Write MSB / Read LSB

/SEL = (PARSEL + IDEADR) * /IORQ * M1               ; Board Access

/CLK = (DATADR + TFRADR) * /IORQ                    ; LH-Clock: Data & Task File

LH := /LH * /TFRADR                                 ; FlipFlop: LSB/MSB Toggle
                                                    ; Reset if Task File Access

WRLO = DATADR * /IORQ * /WR * /LH                   ; write Data LSB in latch
+ (TFRADR + CS1ADR) * /IORQ * /WR                   ; transparent for all others

/WR16 = IDEADR * /IORQ * /WR                        ; MSB and latched LSB to IDE

/RD16 = DATADR * /IORQ * /RD *      LH              ; read Data LSB, latch MSB
 + (TFRADR + CS1ADR) * /IORQ * /RD                  ; all others transparent

/RDHI = DATADR * /IORQ * /RD * /LH                  ; read Data MSB from latch
```

IDE-Interface

220392 TR





IDE

Fun and Games

All Readers

FORTH

8 Queens in Forth

by Frank Sergeant

I need to thank Frank for putting 5 copies of his story on the disk he sent me. The post office managed to mangle the disk so bad that 4 of the 5 stories were unreadable. Only the inner most story was still undamaged. I had to take the envelope appart and use it on the bottom half of the jacket. After all that it made reviewing his article a joy. As always Frank's solutions to age old problems makes reading damaged disk seem like childs play. BDK

It all started when a correspondent mentioned he was converting the code for the 8 Queens problem from Fig Forth to Pygmy Forth and adding trace routines to try to make sense of how it worked. It so happens that the 8 Queens problem had been discussed and assigned a few weeks earlier in the data structures course I was taking at the local university. The course work had to be done in C, but I first worked out the solution in Forth as its interactive nature and quick response make it easy to try out various ideas and approaches. Henry was working with the version of 8 Queens written by Jerry LeVan that appeared in Forth Dimensions vol II, number 1. LeVan's entire solution fit comfortably in only 3 screens. Unfortunately it omitted all comments, even stack comments, and used short but unenlightening array names such as a, b, c, and x. No wonder the algorithm was hard to follow!

Well, I wondered, is it possible to present the 8 Queens solution readably in Forth? See Listing 1 for my attempt.

The problem is to place 8 Queens on an 8 x 8 chessboard so that no Queen is under attack by any other. At first I thought I'd need an 8 x 8 array to represent the board. Each element would be on or off to represent whether a Queen was present on the square. A Queen cannot be placed on a square that is already under attack, so I added an 8-element array to indicate whether any Queen was attacking a particular column, another 8-element array to indicate whether any Queen was attacking a particular row, and two 15-element arrays to indicate whether any Queen was attacking a particular down-sloping or up-sloping diagonal.

After further study I realized I could dispense with the column array. As we place the Queens from left to right, only one Queen at a time in each column, we know there cannot already be a Queen in the column we are about to use. Then I saw how a single 8-element array could represent the board. Each of the 8 elements could represent a column and hold a row number, or it could represent a row, and hold a column number. I decided to let each element represent a row and hold a column number. This makes the array do double duty. It holds the information needed to draw the board and it indicates whether a particular row is under attack.

I saw no way to eliminate the two diagonal arrays. It so happens that all the squares in a given diagonal can be converted to the same number. The row and column numbers of the squares in each down-sloping diagonal have a common difference, e.g.

(1,1), (2,2), (3,3), ... have a difference of 0,
(2,1), (3,2), (4,3), ... have a difference of 1,

while the row and column numbers of the squares in each up-sloping diagonal have a common sum, e.g.

(7,1), (6,2), (5,3), ... sum to 8,
(6,1), (5,2), (4,3), ... sum to 7.

Figure 2 illustrates this, and the last two screens of Listing 1 show the corresponding source code.

There are 15 diagonals in each direction. The phrase - 7 + converts a row and column to an index between 0 and 15 for the down-sloping diagonals. The phrase + 2 - converts a row and column to an index between 0 and 15 for the up-sloping diagonals.

The Alogrithm

There are 8 Queens to place, one in each column. The heart of the solution is the recursive word PLACE. We call it with a row and column number. It checks whether that square is under attack. If so, it simply returns. If that square is not under attack, PLACE marks that square as having a Queen. Then it checks to see whether the column number is 8. If so, we have a solution, because a Queen has been placed successfully in all 8 columns, and this instance of PLACE "prints the board" and cleans up after itself by unmarking the square it just marked. If the column number where it just marked a Queen is not equal to 8, then it calls itself 8 times, once for each of the squares in the next column, and then unmarks the square it just marked. Note that "it" didn't just mark 8 more squares, it only marked one square. "It" did, though, invoke 8 other instances of PLACE, each of which may or may not have marked and unmarked a cell and perhaps invoked an additional 8 instances of PLACE, etc. Is this clear as mud? Let's walk through the source code, maybe it'll get clearer.

Screen 1073 merely loads the code in the other screens. Screen 1074 sets up the arrays. CREATE ROWS 8 ALLOT sets up an 8-byte array named ROWS. When ROWS is later executed, it returns the starting address of this array. : ROW (row - a) 1- ROWS + ; defines the word ROW which takes a single argument, a row number between 1 and 8, and converts it to a number between 0 and 7 by subtracting 1, and adds this offset to the array ROWS. Thus, the address of the byte corresponding to the row number is returned. (''Returned'' means left on the stack; all arguments are passed on the stack.) So, we have a ''smart'' array, or at least ROW is a smart array-handling word, in that it lets us use row numbers 1 through 8 and automatically converts to the proper offsets.

Then (still on screen 1073) the arrays \DIAGS and /DIAGS are created in the same manner, but each is ALLOT'd 15 bytes. Their array handling words, \DIAG and /DIAG are even smarter. They take as arguments row and column numbers and convert them to the proper offset and add that offset to the start of the array. Remember our old friends, the phrases - 7 + and + 2 - ? Note that all three words ROW, \DIAG, and \DIAG return the address of the element we want. It will then be up to our code to do something with that address, such as storing something into it or reading (fetching) the value out of it.

Screen 1075 defines some little helper words. We could do without them, by writing out the insides of their definitions every place we need them, but I think they help make the later code more readable. ERASE takes an address and count and fills that many bytes with zeroes. It is already built into many Forths. ON and OFF take an address of a byte and store either all ones or all zeroes into the byte. (C! takes a value and address and stores the value into the byte at the given address.) ON and OFF are commonly used to work on 16-bit words (2 bytes), but for our convenience we redefine them to work on single bytes. LAST-COLUMN? should be pretty obvious. It takes a column number and returns true if the column number is 8, otherwise it returns false. BUMP-ROW takes a row and column and adds a 1 to the row number without disturbing the column number. Certainly these last two definitions could be left out, but then we might stumble over 1 +UNDER without quite realizing what it is doing, i.e. bumping up the row number. 8 = is clear enough as to what it is doing, but saying instead LAST-COLUMN? reminds us why we are doing it.

On screen 1076, RESET-BOARD takes no arguments and returns none. It fills all three of our arrays with zeroes. MARK takes a row and column as arguments and returns nothing. Note that its definition is divided into 3 separate phrases, with some spacing between them. The spacing (at Wil Baden's suggestion) is usually used to indicate the parameter stack again contains the same items (not necessarily the same values) as the last time the contents were explicitly shown. When MARK is called the stack is expected to contain a row number and a column number. The first phrase makes a copy of these numbers (2DUP duplicates two numbers) and uses up these copies. Thus at the beginning of the second phrase, the stack once again contains the row and column numbers. The first two phrases set the proper element in /DIAGS and \DIAGS on. The third phrase is a little trickier. It converts the row number to the address of the proper element of ROWS and then stores the column number into that element. UNMARK simply undoes everything MARK did.

In screen 1077, MARKED? takes a row and column number and returns true if the byte stored at the row-th element of ROWS matches the column number. In other words, it returns true if a Queen is currently sitting on this square. DANGER? takes a row and column and returns true if that square is under attack. It does it by ORing the proper element of ROWS, \DIAGS, and /DIAGS, which say whether the square is under attack by another Queen in the same row or diagonal. It uses the words PUSH and POP (called >R and R> in some Forths) to move a number off the data stack to the return stack and then back again.

Screen 1078 defines words to print the chessboard. The dot at the beginning the the word name stands for ''print.'' Borders at the top and left show row and column numbers. A ''Q'' or and ''x'' shows that a Queen is or is not present in that square. FOR ... NEXT takes a single argument saying how many times to do the loop. If the number is zero, the loop is not entered at all. CR outputs a carriage return. 4 .R takes a number off the stack and prints it right justified in a 4-character field. The body of an IF ... THEN is done if the flag on the stack is non-zero. EMIT prints one character. In .SQUARE, EMIT prints either a ''Q'' or an ''x'' and the ''c'' in the stack comment just before EMIT stands for ''character,'' even though I've often used ''r c'' to stand for ''row column.''

In screen 1079, PRINT-BOARDS counts the number of solutions we have found, and prints out the first few (based on the value of the constant #BOARDS-TO-PRINT). 1 #BOARDS +! increments the value of the variable #BOARDS. #BOARDS @ puts the contents of the variable #BOARDS on the stack, and U. prints that value as an unsigned integer. ?SCROLL tests the keyboard and pauses the display if a keypress is waiting. In Pygmy you can stab out blindly and stop the scrolling by pressing any key, as long as the loop contains ?SCROLL. Pressing another key restarts the display. Pressing Esc aborts the execution altogether.

All that is easy enough, but we have two words left. Look at the definition of 8QUEENS first. This is the highest level word in the application, and what you actually type from the keyboard to get the solution. It resets the board to make sure the arrays start out empty. It stores a zero into the variable #BOARDS. Then it calls PLACE 8 times, once for each square in the first column. PLACE will take care of calling PLACE for all the other columns. Finally, look at the definition of PLACE in screen 1080 and compare it to the discussion earlier about what PLACE should do. Note the use of the word PLACE (on line 12 of screen 1080) inside the definition of PLACE. Here we want the word PLACE to invoke another instance of itself, rather

than invoking some other previously defined word that just happened to be named PLACE. The word RECURSIVE fixes up the word we are defining so it can call itself. The RECURSIVE is only active at compile time and does not become part of the final definition. RECURSIVE could have been placed anywhere in the definition prior to the calling of PLACE. Some Forths omit the word RECURSIVE and use the word MYSELF or RECURSE instead of PLACE to accomplish the same thing. Listing 3 shows the output produced by 8QUEENS.

There you have it. Is it readable? I enjoyed working it up in Forth before having to program it in C. It was pleasant to be able to test the individual words, and inspect the contents of variables and arrays, all from the keyboard.

Among other things, Frank makes EPROM programmers affordable by everyone. If you would like a flier about his Bare Bones EPROM Programmer Kit, from plans, to partial kit, to fully assembled, send him a SASE at 809 W. San Antonio Street, San Marcos, Texas 78666.

Figure 1. The source code in Pygmy Forth for 8 Queens.

scr # 1073
```
( 8 QUEENS  LOAD SCREEN )

1074 1081 THRU   ( the basic 8 Queens solution)

1082 1083 THRU   ( proof that squares in the same
diagonal share the same diagonal number)
```

scr # 1074
```
( 8 Queens   - arrays for rows and diagonals)
 CREATE ROWS  8 ALLOT
: ROW ( row - a)  1- ROWS +  ;
                        ( row element contains col#)

 CREATE \DIAGS 15 ALLOT
: \DIAG ( r c - a)  - 7 +  \DIAGS +  ;  ( either on or off)

 CREATE /DIAGS 15 ALLOT
: /DIAG ( r c - a)  + 2 -  /DIAGS +  ; ( either on or off)
```

scr # 1075
```
( 8 Queens -  some little helpers)

: ERASE ( a # -)   0 FILL ;

: ON  ( a -) -1 SWAP C! ;   ( to turn bytes on & off instead )
: OFF ( a -)  0 SWAP C! ;
                        (   of words, which is more usual)

: LAST-COLUMN?  ( column - f)  8 =  ;

: BUMP-ROW ( r c - r+1 c)   1 +UNDER  ;
```

scr # 1076
```
( Each square is in one row and two diagonals)
 : RESET-BOARD ( -)
  ROWS 8 ERASE   \DIAGS 15 ERASE
                 /DIAGS 15 ERASE  ,

: MARK ( r c -)  2DUP \DIAG ON   2DUP
                 /DIAG ON   SWAP ROW  C! ;

: UNMARK ( r c -) OVER ROW OFF   2DUP
                 \DIAG OFF    /DIAG OFF  ;
```

scr # 1077
```
( 8 Queens - check the square to see if it is under attack)
: MARKED? ( r c - f)  SWAP ROW C@ =  ;


: DANGER? ( r c - f)
   OVER ROW C@ PUSH
   2DUP /DIAG C@ PUSH   \DIAG C@
                  POP POP OR OR  ;
```

scr # 1078
```
( 8 Queens - print the board )
: .COLUMN-HEADINGS ( -)
   CR ."      1  2  3  4  5  6  7  8"  ;

: .ROW-HEADING ( row -)  4 .R  ;

: .SQUARE ( row col -)
   3 SPACES    MARKED? ( f) IF 'Q ELSE 'x
    THEN ( c) EMIT ;

: .BOARD ( -)
  .COLUMN-HEADINGS
   1 1 8 FOR ( r c)  CR    OVER .ROW-HEADING
                2DUP ( r c r c)
       8 FOR   2DUP .SQUARE    1+ NEXT    2DROP ( r c)
        BUMP-ROW ( r c)
       NEXT  2DROP ;
```

scr # 1079
```
( 8 Queens -  count each solution and optionally print it)
VARIABLE #BOARDS

5 CONSTANT #BOARDS-TO-PRINT

: PRINT-BOARD ( -)
  1 #BOARDS +!
  #BOARDS @ #BOARDS-TO-PRINT > NOT ( f)
  IF ( )  CR CR ." Solution #"   #BOARDS @ U.   .BOARD
THEN
  ?SCROLL  ;
```

scr # 1080
```
( 8 Queens - the recursive heart of it all)
: PLACE ( row col -)   RECURSIVE
  2DUP    DANGER?
  IF 2DROP  ( ) ( ie bail out, we've hit a dead end)
  ELSE ( r c) 2DUP MARK
    DUP LAST-COLUMN?
    IF ( r c) PRINT-BOARD   UNMARK ( )
                ( ie bail out, success)
    ELSE ( r c) ( now try every row in the next column)
      1 OVER 1+ ( ie newrow=1 & newcol = c+1)
      8 FOR ( r c  r' c')
        2DUP PLACE    BUMP-ROW ( r c r' c')
       NEXT 2DROP  ( r c)  UNMARK (  )
    THEN
  THEN  ;
```

scr # 1081
```
( 8 Queens - put it all together)
: 8QUEENS ( -)
  RESET-BOARD    0 #BOARDS !
  1 1  ( r c) ( ie starting at this square,)
  8 FOR ( r c) 2DUP PLACE    BUMP-ROW
   NEXT   2DROP ( )
  CR CR ." there were " #BOARDS @ U. ." solutions"
      CR  ;
  ( try placing a queen in each row of the first column.
    Whenever PLACE is successful, it pushes its luck by
    calling itself again to try to place a queen in each row
    of the next column, which when it is successful .... )
```

scr # 1082
```
( print diagonal values of each square in a given row)
: .RowNegDiags ( row -)
   CR DUP  4 .R  ( ie print the row# at left border of board)
   1  8 FOR ( row col) 2DUP  - 7 +  4 .R    1+ NEXT
2DROP ;


: .RowPosDiags ( row -)
   CR DUP  4 .R  ( ie print the row# at left border of board)
   1  8 FOR ( row col) 2DUP  + 2 -  4 .R
    1+ NEXT 2DROP ;
```

scr # 1083
```
( print diagonal values to demonstrate that all cells in the
  same diagonal have the same value )
: .NEG ( -)  CR ."  Negative Sloping Diagonals,  r c - 7 +"
  .COLUMN-HEADINGS
   1  8 FOR ( row)  DUP .RowNegDiags  1+ NEXT DROP ;

: .POS ( -)  CR ."  Positive Sloping Diagonals,  r c + 2 -"
  .COLUMN-HEADINGS
   1  8 FOR ( row)  DUP .RowPosDiags  1+ NEXT DROP ;

: .BOTH  CR .NEG CR CR .POS CR ;
```

Figure 2. The chessboard diagonal numbers.
```
Negative Sloping Diagonals,  r c - 7 +
    1  2  3  4  5  6  7  8
 1  7  6  5  4  3  2  1  0
 2  8  7  6  5  4  3  2  1
 3  9  8  7  6  5  4  3  2
 4 10  9  8  7  6  5  4  3
 5 11 10  9  8  7  6  5  4
 6 12 11 10  9  8  7  6  5
 7 13 12 11 10  9  8  7  6
 8 14 13 12 11 10  9  8  7
Positive Sloping Diagonals,  r c + 2 -
    1  2  3  4  5  6  7  8
 1  0  1  2  3  4  5  6  7
 2  1  2  3  4  5  6  7  8
 3  2  3  4  5  6  7  8  9
 4  3  4  5  6  7  8  9 10
 5  4  5  6  7  8  9 10 11
 6  5  6  7  8  9 10 11 12
 7  6  7  8  9 10 11 12 13
 8  7  8  9 10 11 12 13 14
```

Figure 3. The final program output showing the first five solutions.
```
Solution #1
    1  2  3  4  5  6  7  8
 1  Q  x  x  x  x  x  x  x
 2  x  x  x  x  x  x  Q  x
 3  x  x  x  x  Q  x  x  x
 4  x  x  x  x  x  x  x  Q
 5  x  Q  x  x  x  x  x  x
 6  x  x  x  Q  x  x  x  x
 7  x  x  x  x  x  Q  x  x
 8  x  x  Q  x  x  x  x  x
Solution #2
    1  2  3  4  5  6  7  8
 1  Q  x  x  x  x  x  x  x
 2  x  x  x  x  x  x  Q  x
 3  x  x  x  Q  x  x  x  x
 4  x  x  x  x  x  Q  x  x
 5  x  x  x  x  x  x  x  Q
 6  x  Q  x  x  x  x  x  x
 7  x  x  x  x  Q  x  x  x
 8  x  x  Q  x  x  x  x  x
Solution #3
    1  2  3  4  5  6  7  8
 1  Q  x  x  x  x  x  x  x
 2  x  x  x  x  x  Q  x  x
 3  x  x  x  x  x  x  x  Q
 4  x  x  Q  x  x  x  x  x
 5  x  x  x  x  x  x  Q  x
 6  x  x  x  Q  x  x  x  x
 7  x  Q  x  x  x  x  x  x
 8  x  x  x  x  Q  x  x  x
Solution #4
    1  2  3  4  5  6  7  8
 1  Q  x  x  x  x  x  x  x
 2  x  x  x  x  Q  x  x  x
 3  x  x  x  x  x  x  x  Q
 4  x  x  x  x  x  Q  x  x
 5  x  x  Q  x  x  x  x  x
 6  x  x  x  x  x  x  Q  x
 7  x  Q  x  x  x  x  x  x
 8  x  x  x  Q  x  x  x  x
Solution #5
    1  2  3  4  5  6  7  8
 1  x  x  x  x  x  Q  x  x
 2  Q  x  x  x  x  x  x  x
 3  x  x  x  x  Q  x  x  x
 4  x  Q  x  x  x  x  x  x
 5  x  x  x  x  x  x  x  Q
 6  x  x  Q  x  x  x  x  x
 7  x  x  x  x  x  x  Q  x
 8  x  x  x  Q  x  x  x  x
```

there were 92 solutions

Real Computing

By Rick Rodman

32-Bit Systems

All Readers

BSD 386, MINIX

Thanks to RickI don't talk about MINIX in THE COMPUTER CORNER these days. He has a much better handle on the 32-bit world than I do. I keep thinking about getting a real computer, but not sure which one to get. This time Rick gives us the latest news, saving the "what to get" for a later issue. BDK.

Linux and BSD 386

Linus Torvalds of Finland has released the source for a 386-specific clone of the BSD Unix kernel, which he calls Linux. This is available by anonymous ftp from various places, which is to say I don't have a copy yet. Also, a "bootable" version of BSD386, a non-AT&T ver sion of real BSD for the 386 or 486, has been released by Berkeley for ftp access. There are some folks working on porting each of these operating systems to the PC-532.

What's going on with Mach, you ask? My, we live in interesting times indeed.

Embeddable CP/M

Years ago, a friend of mine put a row of 2708 EPROMs on his homebrew computer with a complicated counter circuit. These EPROMs contained the CP/M CCP, BDOS, and BIOS. On cold and warm boots, CP/M was loaded from these EPROMs into RAM. While it still had to log in the A: device, the speed of this system was quite remarkable.

EPROMs are available today in capacities of 512 kbytes per chip - for far less than we paid for 2708s in those days. Why not put Z-system in a chip? Tell system designers where in the chip to put their BIOS, and put the whole shmear in EPROM? Cheap PROM burners are limited to 27256s or 27128s, so it'd be better to use one of these sizes.

The reason for the clumsy counter circuit was because the CCP and BDOS maintain inter nal variables within themselves. To make the Z-system run from PROM, it should move these scratchpad variables into the base page of the system somewhere. As I recall, there were very few of these for CP/M; I fiddled at one time with changing them to point to RAM and running CP/M from a Bytesaver II many years back.

Protecting Idiots from Themselves

Both DR-DOS 6 and MS-DOS 5 have added a bunch of "idiot protection" stuff in them. This is the very stuff which makes technically aware users less and less happy with DOS. The worst is what they've done to the FORMAT command, which won't really format the disk unless you add a "/u" option - and sometimes won't even do it then!

In my experience, it's a safety procedure to reformat floppy disks when I re-use them. This refreshes the magnetic fields of the ID marks and ensures that the whole disk is usable. With these new DOSs, reformatting is a real pain. Another thing that's gone is using DISKCOPY's "formatting while copying". While they may say it still works, in practice it doesn't. You have to format all the target disks first.

What in the world is the industry thinking with respect to DOS? That it's only for stupid users? Neither OS/2 1.3 nor 2.0 has any of this "stupid user" logic.

Minix News

Andy Tanenbaum has officially released new compilers for Minix in a separate package. ANSI C, Pascal, and Modula-2 compilers are included, in object code form only. Andy claims that the source code would be of no interest, as it was "generated" with the Amster dam Compiler Kit. (Maybe I'm old-fashioned, but to me, "source code" is by definition the o73 original form of the code, that which the human programmer wrote; anything "generated" is "source" but "object".)

There was some discussion in which the compilability of Minix 2.0 appears to come into question. First, Andy wrote: "It is my intention to go over to the ANSI C compiler for MINIX 2.0. In fact, I have been running nothing else but the ANSI compiler for over a year, and it seems pretty solid." This new compiler will not come with Minix, even in binary form. However, Andy also says: "Most of version 1.6.18 is already ANSI-fied. Never theless, as much as possible, I will keep the 2.0 code such that it can be compiled with the old compiler, so you don't have to switch to the new ANSI compiler if you can't afford it." Note the use of the slippery phrase, "as much as possible". The size of the grain of salt necessary is left to the reader's imagination.

The Modula-2 compiler also includes a "makefile generator". The package is available from two companies, which are listed at the end of this column, for about $200.

The shocking rumor has been confirmed: Minix for the Atari ST and for

the IBM PC are not file-system compatible. It is not possible to exchange Minix disks between the two. The reason for this is that, while the 8088 and 68000 are known to use different byte orders (the 8088 is "little endian" and the 68000 is "big endian"), Andy Tanenbaum and his pro grammers made no effort at all to standardize on a byte order. Thus, a historic opportunity for cross-platform portability was carelessly discarded.

Minix experimenters are put on notice that Andy Tanenbaum, who has said many times that he regards Minix as a "hobby", has little concern for their attempts to do real work with this operating system. I'm personally very disappointed, because Minix is, after all, one of the most expensive PC operating systems available.

In happier news, I recently added a SCSI floppy to my PC-532, and used it to transfer a great many files. The PC-532 is compatible with the PC Minix file system and can ex change disks with PCs running Minix. Also, the MIN2DOS package, which allows reading and writing Minix floppies from the PC, can be used to transfer data from DOS to a Minix floppy.

A SCSI floppy? Yes, they are available. This one is a TEAC FC-1-01, and can read and write any 3-1/2" format up to the 3.88-megabyte "ED" capacity. It's available for about $200 from J. D. Hannam. ED diskettes at my local CompUSA sell for a stratospheric $80 per box, so I won't be buying any of those real soon.

Free Ideas

Here are some ideas that'll make somebody lots of money. (If it's you, remember where you read them.) In the past, you had to get something to work to get a patent, but the Gilbert Hyatt and RSA cases show that you can get a patent today for almost anything, just on the basis of chicken scratches on the back of an envelope. I don't agree with this, so I'm not patenting them, but if someone does, this article will constitute equivalent prior art. Anyway, these are products I would expect to see soon, probably all before the end of 1993.

3-inch CD-ROM: CD-ROM will really hit stride as a software distribution medium when 3" CD-ROMs come out. The 5-inch CD-ROMs are just too big, both physically and in byte capacity (600 megabytes). While CD3 audio disks were a flop, the computer industry is different. Another factor will be Kodak's Photo CD product, which will drastically reduce the cost of producing CD-ROMs in small quantities. CD3-ROMs, with 160 to 200 mega bytes, will play in your regular CD-ROM drive with an adapter.

Portable teleprompter: This would be just the thing for people who are giving presentations or sales pitches. While electronic organizers are popular now, none have this feature. The display should "crawl" with smooth scroll at a slow rate, but allow the speaker to discreetly stop the scrolling, or scroll forward or back.

Self-aligning printer: By adding inexpensive optical sensors to a printer, the printer can recognize lines or boxes on a printed form and automatically align to the form for printing. This fits into the major trend toward smarter printers.
Printed food: Computer-driven systems with CCD cameras for automatic alignment will spray precisely-patterned food coloring on mass-produced food items, producing snack foods and other items with logos of sports teams and popular entertainment stars, as well as advertising. As this industry matures, photograph-like images will become possible. This will be a major fad at first, but become commonplace as time goes on.

Personally, I find the idea of printed food unappetizing, but given recent history, you have to admit it's inevitable. Virtually everything is an "advertising vehicle" these days, and people love it - they actually pay for the privilege of advertising sports teams and rock stars by buying T-shirts and other stuff at premium prices. Printed-food technology will be quite clever and involve major investments in hardware and engineering talent - which could mean you.

How about it, readers? Am I all wet? Maybe I can get the Editor's permission to send a sporty TCJ Coffee Mug (or a bag of TCJ Potato Chips?) to the first reader to spot a commercial implementation of any of these ideas. Have more ideas along these lines? Send 'em in, and see your name in genuine print!

Next Time

Next time we'll discuss IBM's OS/2 2.0 - from an experimenter's perspective. For example, I'll describe how to run Minix within OS/2. Plus I'll have more insight on using the SCSI bus in new and wondrous ways. 'Til then, remember, you don't need windows when it's warm and sunny.

Where to call or write:

SCSI Floppy:

J. D. Hannam
1182 North Grove Unit C
Anaheim, CA 92806
1-800-228-0308

Minix compilers:

Transmediair Products & Support B.V.
Melkweg 3
3721 RG Bilthoven
The Netherlands
Tel: +31 30 281820
FAX: +31 30 292294

Unipress Software
2025 Lincoln Highway
Edison, NJ 08817 U.S.A.
Tel: +1 908 287 2100
FAX: +1 908 287 4929
Email: msk@unipress.com

NEW ADDRESS
The Computer Journal
P. O. Box 535
Lincoln, CA
95648-0535
1-(916) 645-1670

Kaypro-84 Direct File Transfers Without Null Modem

By WalterJ. Rottenkolber

8-Bit Systems

Intermediate

Data - I/O

Although this is a mostly Kaypro upgrading project, I was interested in the simplicity of the project. All too often we pass over the easiest way for a more complex or traditional answer. It got me thinking and I hope it stirs your own little gray cells into some work. BDK

If your Kaypro is one of the 1984 or newer versions, you can do direct computer to computer file transfers without needing a null-modem cable or adapter. These Kaypros have two serial ports -- modem and printer -- configured differently. The modem port is wired as Data Terminal Equipment (DTE), whereas the printer port is setup as Data Communications Equipment (DCE). As a result, the printer port of one computer can be connected to the modem port of another with an ordinary RS232-C serial cable.

The only catch is that modem programs are (naturally) configured for the modem port. So you need to redesign a modem program to operate through the printer port. This is easily done on the modem configuration overlay because only the baud port, and the SIO data and control port addresses need to be changed.

Since the same type of chips are used for both ports, no changes are necessary for register numbers, control codes, and bit masks. Fig. 1 shows the SIO port addresses, and Fig. 2 is an abbreviated example of the changes to the IMP245 Kaypro overlay.

I found that MEX114 could handle up to 4800 b/s reliably in direct file transfers, but that IMP245 could go up another notch to 9600 b/s without choking. Seeing those kilobytes flash by made IMP the best choice.One peculiarity of IMP shows itself in batch mode. Searching for filenames and placing them in the name array takes an unexpectedly long time. If the batch has more than about seven file names, the process can take long enough for the receiving system to timeout. In one batch of 36 files, it took almost four minutes to process the filenames before transfers began.

Unfortunately, no message alerts you when filename processing is over. You will simply have to find by experiment the time to delay before starting the batchfile receive.

FIG. 1
Port Addresses

Modem (DTE)		Printer (DCE)
Baud	-- 00H	0BH
SIO Data	-- 04H	0CH
SIO Ctl/St	-- 06H	0EH

FIG. 2
MP245 Kaypro Overlay Changes

```
PORT        EQU   0CH     ; Your base port (data or status)
MDCTL1      EQU   PORT+2          ; Modem control port
MDDATP      EQU   PORT    ; Modem data port
MDRCV       EQU   01H     ; Modem receive ready
MDSND       EQU   04H     ; Modem send ready bit
MDTXE       EQU   01H     ;Modem send buffer empty, holding buffer empty
;
BRPORT      EQU   0BH     ; Baud rate generator port
;
; These routines and equates are at the beginning of the program so
; they can be patched by a monitor or overlay file without re-assembling
; the program.
;
MSPEED:       DB    8       ; 0=110 1=300 2=450 3=600 4=710 5=1200
                            ; 6=2400 7=4800 8=9600 9=19200 default
HS2400: DB    YES           ; Yes=2400 bps highest speed
HS1200: DB    NO            ; Yes=1200 bps highest speed
;
LOGPTR:       DW    LOGON           ; Pointer to display LOGON message
;
SYSVR:  CALL J$ILPRT                ; Display the following line
  DB    'Version for Kaypro computers, Printer serial port' ;1AAH
  DB    ' (0CH)',CR,LF,0
  RET
```

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Special Close Out Sale on these back issues only.

3 or more, $1.50 each postpaid in the US or $3.00 postpaid airmail outside US.

Issue Number 1:
- RS-232 Interface Part 1
- Telecomputing with the Apple II
- Beginner's Column: Getting Started
- Build an "Epram"

Issue Number 2:
- File Transfer Programs for CP/M
- RS-232 Interface Part 2
- Build Hardware Print Spooler Part 1
- Review of Floppy Disk Formats
- Sending Morse Code with an Apple II
- Beginner's Column: Basic Concepts and Formulas

Issue Number 3:
- Add an 8087 Math Chip to Your Dual Processor Board
- Build an A/D Converter for Apple II
- Modems for Micros
- The CP/M Operating System
- Build Hardware Print Spooler: Part 2

Issue Number 4:
- Optronics, Part 1: Detecting, Generating and Using Light in Electronics
- Multi-User: An Introduction
- Making the CP/M User Function More Useful
- Build Hardware Print Spooler: Part 3
- Beginner's Column: Power Supply Design

Issue Number 8:
- Build VIC-20 EPROM Programmer
- Multi-User: CP/Net
- Build High Resolution S-100 Graphics Board, Part 3
- System Integration, Part 3: CP/M 3.0
- Linear Optimization with Micros

Issue Number 18:
- Parallel Interface for Apple II Game Port
- The Hacker's MAC: A Letter from Lee Felsenstein
- S-100 Graphics Screen Dump
- The LS-100 Disk Simulator Kit
- BASE: Part Six
- Interfacing Tips & Troubles: Communicating with Telephone Tone Control, Part 1

Issue Number 19: Sold Out
- Using the Extensibility of Forth
- Extended CBIOS
- A $500 Superbrain Computer
- BASE: Part 7
- Interfacing Tips & Troubles: Communicating with Telephone Tone Control, Part 2
- Multitasking & Windows with CP/M: A Review of MTBASIC

Issue Number 20:
- Designing an 8035 SBC
- Using Apple Graphics from CP/M: Turbo Pascal Controls Apple Graphics
- Soldering & Other Strange Tales
- Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K

Issue Number 21:
- Extending Turbo Pascal: Customize with Procedures & Functions
- Unsoldering: The Arcane Art
- Analog Data Acquisition & Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC

Issue Number 22:
- NEW-DOS: Write Your Own Operating System
- Variability in the BDS C Standard Library
- The SCSI Interface: Introductory Column
- Using Turbo Pascal ISAM Files
- The Ampro Little Board Column

Issue Number 23:
- C Column: Flow Control & Program Structure
- The Z Column: Getting Started with Directories & User Areas
- The SCSI Interface: Introduction to SCSI
- NEW-DOS: The Console Command Processor
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program to Create an Index
- The Ampro Little Board Column

Issue Number 24:
- Selecting & Building a System
- The SCSI Interface: SCSI Command Protocol
- Introduction to Assemble Code for CP/M
- The C Column: Software Text Filters
- Ampro 186 Column: Installing MS-DOS Software
- The Z-Column
- NEW-DOS: The CCP Internal Commands
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Issue Number 25:
- Repairing & Modifying Printed Circuits
- Z-Com vs. Hacker Version of Z-System
- Exploring Single Linked Lists in C
- Adding Serial Port to Ampro LB
- Building a SCSI Adapter
- NEW-DOS: CCP Internal Commands
- Ampro 186 Networking with SuperDUO
- ZSIG Column

Issue Number 26:
- Bus Systems: Selecting a System Bus
- Using the SB180 Real Time Clock
- The SCSI Interface: Software for the SCSI Adapter
- Inside Ampro Computers
- NEW-DOS: The CCP Commands (continued)
- ZSIG Corner
- Affordable C Compilers
- Concurrent Multitasking: A Review of DoubleDOS

Issue Number 27:
- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
- The Art of Source Code Generation: Disassembling Z-80 Software
- Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation
- The C Column: A Graphics Primitive Package
- The Hitachi HD64180: New Life for 8-bit Systems
- ZSIG Corner: Command Line Generators and Aliases
- A Tutor Program in Forth: Writing a Forth Tutor in Forth
- Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats

Issue Number 28:
- Starting Your Own BBS
- Build an A/D Converter for the Ampro Little Board
- HD64180: Setting the Wait States & RAM Refresh using PRT & DMA
- Using SCSI for Real Time Control
- Open Letter to STD Bus Manufacturers
- Patching Turbo Pascal
- Choosing a Language for Machine Control

Issue Number 29:
- Better Software Filter Design
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1
- Using the Hitachi hd64180: Embedded Processor Design
- 68000: Why use a new OS and the 68000?
- Detecting the 8087 Math Chip
- Floppy Disk Track Structure
- The ZCPR3 Corner

Issue Number 30:
- Double Density Floppy Controller
- ZCPR3 IOP for the Ampro Little Board
- 3200 Hackers' Language
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2
- Non-Preemptive Multitasking
- Software Timers for the 68000
- Lilliput Z-Node
- The ZCPR3 Corner
- The CP/M Corner

Issue Number 31:
- Using SCSI for Generalized I/O
- Communicating with Floppy Disks: Disk Parameters & their variations
- XBIOS: A Replacement BIOS for the SB180
- K-OS ONE and the SAGE: Demystifying Operating Systems
- Remote: Designing a Remote System Program
- The ZCPR3 Corner: ARUNZ Documentation

Issue Number 32:
- Language Development: Automatic Generation of Parsers for Interactive Systems
- Designing Operating Systems: A ROM based OS for the Z81
- Advanced CP/M: Boosting Performance
- Systematic Elimination of MS-DOS Files: Part 1, Deleting Root Directories & an In-Depth Look at the FCB
- WordStar 4.0 on Generic MS-DOS Systems: Patching for ASCII Terminal Based Systems
- K-OS ONE and the SAGE: System Layout and Hardware Configuration
- The ZCPR3 Corner: NZCOM and ZCPR34

Issue Number 33:
- Data File Conversion: Writing a Filter to Convert Foreign File Formats
- Advanced CP/M: ZCPR3PLUS & How to Write Self Relocating Code
- DataBase: The First in a Series on Data Bases and Information Processing
- SCSI for the S-100 Bus: Another Example of SCSI's Versatility
- A Mouse on any Hardware: Implementing the Mouse on a Z80 System
- Systematic Elimination of MS-DOS Files: Part 2, Subdirectories & Extended DOS Services
- ZCPR3 Corner: ARUNZ Shells & Patching WordStar 4.0

Issue Number 34:
- Developing a File Encryption System.
- Database: A continuation of the data base primer series.
- A Simple Multitasking Executive: Designing an embedded controller multitasking executive.
- ZCPR3: Relocatable code, PRL files, ZCPR34, and Type 4 programs.
- New Microcontrollers Have Smarts: Chips with BASIC or Forth in ROM are easy to program.
- Advanced CP/M: Operating system extensions to BDOS and BIOS, RSXs for CP/M 2.2.
- Macintosh Data File Conversion in Turbo Pascal.
- The Computer Corner

Issue Number 35:
- All This & Modula-2: A Pascal-like alternative with scope and parameter passing.
- A Short Course in Source Code Generation: Disassembling 8088 software to produce modifiable assem. source code.
- Real Computing: The NS32032.
- S-100: EPROM Burner project for S-100 hardware hackers.
- Advanced CP/M: An up-to-date DOS, plus details on file structure and formats.
- REL-Style Assembly Language for CP/M and Z-System. Part 1: Selecting your assembler, linker and debugger.
- The Computer Corner

Issue Number 36:
- Information Engineering: Introduction.
- Modula-2: A list of reference books.
- Temperature Measurement & Control: Agricultural computer application.
- ZCPR3 Corner: Z-Nodes, Z-Plan, Amstrand computer, and ZFILE.
- Real Computing: NS32032 hardware for experimenter, CPUs in series, software options.
- SPRINT: A review.
- REL-Style Assembly Language for CP/M & ZSystems, part 2.
- Advanced CP/M: Environmental programming.
- The Computer Corner.

Issue Number 37:
- C Pointers, Arrays & Structures Made Easier: Part 1, Pointers.
- ZCPR3 Corner: Z-Nodes, patching for NZCOM, ZFILER.
- Information Engineering: Basic Concepts: fields, field definition, client worksheets.
- Shells: Using ZCPR3 named shell variables to store date variables.
- Resident Programs: A detailed look at TSRs & how they can lead to chaos.
- Advanced CP/M: Raw and cooked console I/O.
- Real Computing: The NS 32000.
- ZSDOS: Anatomy of an Operating System: Part 1.
- The Computer Corner.

Issue Number 38:
- C Math: Handling Dollars and Cents With C.
- Advanced CP/M: Batch Processing and a New ZEX.
- C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.
- The Z-System Corner: Shells and ZEX, new Z-Node Central, system security under Z-Systems.
- Information Engineering: The portable Information Age.
- Computer Aided Publishing: Introduction to publishing and Desk Top Publishing.
- Shells: ZEX and hard disk backups.
- Real Computing: The National Semiconductor NS320XX.
- ZSDOS: Anatomy of an Operating System, Part 2.

Issue Number 39:
- Programming for Performance: Assembly Language techniques.
- Computer Aided Publishing: The Hewlett Packard LaserJet.
- The Z-System Corner: System enhancements with NZCOM.
- Generating LaserJet Fonts: A review of Digi-Fonts.
- Advanced CP/M: Making old programs Z-System aware.
- C Pointers, Arrays & Structures Made Easier: Part 3: Structures.
- Shells: Using ARUNZ alias with ZCAL.
- Real Computing: The National

Semiconductor NS320XX.
· The Computer Corner.

Issue Number 40:

· Programming the LaserJet: Using the escape codes.
· Beginning Forth Column: Introduction.
· Advanced Forth Column: Variant Records and Modules.
· LINKPRL: Generating the bit maps for PRL files from a REL file.
· WordTech's dBXL: Writing your own custom designed business program.
· Advanced CP/M: ZEX 5.0×The machine and the language.
· Programming for Performance: Assembly language techniques.
· Programming Input/Output With C: Keyboard and screen functions.
· The Z-System Corner: Remote access systems and BDS C.
· Real Computing: The NS320XX
· The Computer Corner.

Issue Number 41:

· Forth Column: ADTs, Object Oriented Concepts.
· Improving the Ampro LB: Overcoming the 88Mb hard drive limit.
· How to add Data Structures in Forth
· Advanced CP/M: CP/M is hacker's haven, and Z-System Command Scheduler.
· The Z-System Corner: Extended Multiple Command Line, and aliases.
· Programming disk and printer functions with C.
· LINKPRL: Making RSXes easy.
· SCOPY: Copying a series of unrelated files.
· The Computer Corner.

Issue Number 42:

· Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
· Using BYE with NZCOM.
· C and the MS-DOS Screen Character Attributes.
· Forth Column: Lists and object oriented Forth.
· The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
· 68705 Embedded Controller Application: An example of a single-chip microcontroller application.
· Advanced CP/M: PluPerfect Writer and using BDS C with REL files.
· Real Computing: The NS 32000.
· The Computer Corner

Issue Number 43:

· Standardize Your Floppy Disk Drives.
· A New History Shell for ZSystem.
· Heath's HDOS, Then and Now.
· The ZSystem Corner: Software update service, and customizing NZCOM.
· Graphics Programming With C: Graphics routines for the IBM PC, and the Turbo C graphics library.
· Lazy Evaluation: End the evaluation as soon as the result is known.
· S-100: There's still life in the old bus.
· Advanced CP/M: Passing parameters, and complex error recovery.
· Real Computing: The NS32000.
· The Computer Corner.

Issue Number 44:

· Animation with Turbo C Part 1: The Basic Tools.
· Multitasking in Forth: New Micros F68FC11 and Max Forth.
· Mysteries of PC Floppy Disks Revealed: FM, MFM, and the twisted cable.
· DosDisk: MS-DOS disk format emulator for CP/M.
· Advanced CP/M: ZMATE and using lookup and dispatch for passing parameters.
· Real Computing: The NS32000.
· Forth Column: Handling Strings.
· Z-System Corner: MEX and telecommunications.
÷ The Computer Corner

Issue Number 45:

· Embedded Systems for the Tenderfoot: Getting started with the 8031.
· The Z-System Corner: Using scripts with MEX.
· The Z-System and Turbo Pascal: Patching TURBO.COM to access the Z-System.
· Embedded Applications: Designing a Z80 RS-232 communications gateway, part 1.
· Advanced CP/M: String searches and tuning Jetfind.
· Animation with Turbo C: Part 2, screen interactions.
· Real Computing: The NS32000.
· The Computer Corner.

Issue Number 46:

· Build a Long Distance Printer Driver.
· Using the 8031's built-in UART for serial communications.
· Foundational Modules in Modula 2.
· The Z-System Corner: Patching The Word Plus spell checker, and the ZMATE macro text editor.

The Computer Journal

Back Issues

Sales limited to supplies in stock.

· Animation with Turbo C: Text in the graphics mode.
· Z80 Communications Gateway: Prototyping, Counter/Timers, and using the Z80 CTC.

Issue Number 47:

· Controlling Stepper Motors with the 68HC11F
· Z-System Corner: ZMATE Macro Language
· Using 8031 Interrupts
· T-1: What it is & Why You Need to Know
· ZCPR3 & Modula, Too
· Tips on Using LCDs: Interfacing to the 68HC705
· Real Computing: Debugging, NS32 Multitasking & Distributed Systems
· Long Distance Printer Driver: correction
· ROBO-SOG 90
· The Computer Corner

Issue Number 48:

· Fast Math Using Logarithms
· Forth and Forth Assembler
· Modula-2 and the TCAP
· Adding a Bernoulli Drive to a CP/M Computer (Building a SCSI Interface)
· Review of BDS "Z"
· PMATE/ZMATE Macros, Pt. 1
· Real Computing
· Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX
· Z-Best Software
· The Computer Corner

Issue Number 49:

· Computer Network Power Protection
· Floppy Disk Alignment w/RTXEB, Pt. 1
· Motor Control with the F68HC11
· Controlling Home Heating & Lighting, Pt. 1
· Getting Started in Assembly Language
· LAN Basics
· PMATE/ZMATE Macros, Pt. 2
· Real Computing
· Z-System Corner
· Z-Best Software
· The Computer Corner

Issue Number 50:

· Offload a System CPU with the Z181
· Floppy Disk Alignment w/RTXEB, Pt. 2
· Motor Control with the F68HC11
· Modula-2 and the Command Line
· Controlling Home Heating & Lighting, Pt. 2
· Getting Started in Assembly Language Pt 2
· Local Area Networks
· Using the ZCPR3 IOP
· PMATE/ZMATE Macros, Pt. 3
· Z-System Corner, PCED
· Z-Best Software
· Real Computing, 32FX16, Caches
· The Computer Corner

Issue Number 51:

· Introducing the YASBEC
· Floppy Disk Alignment w/RTXEB, Pt 3
· High Speed Modems on Eight Bit Systems
· A Z8 Talker and Host
· Local Area Networks--Ethernet
· UNIX Connectivity on the Cheap
· PC Hard Disk Partition Table
· A Short Introduction to Forth
· Stepped Inference as a Technique for Intelligent Real-Time Embedded Control
· Real Computing, the 32CG160, Swordfish, DOS Command Processor
· PMATE/ZMATE Macros
· Z-System Corner, The Trenton Festival
· Z-Best Software, the Z3HELP System
· The Computer Corner

Issue Number 52:

· YASBEC, The Hardware
· An Arbitrary Waveform Generator, Pt. 1
· B.Y.O. Assembler...in Forth
· Getting Started in Assembly Language, Pt. 3
· The NZCOM IOP
· Servos and the F68HC11
· Z-System Corner, Programming for Compatibility
· Z-Best Software
· Real Computing, X10 Revisited
· PMATE/ZMATE Macros
· Controlling Home Heating & Lighting, Pt. 3
· The CPU280, A High Performance Single-Board Computer
· The Computer Corner

	U.S.	Foreign (Surface)	Foreign (Airmail)	Total
Subscriptions				
1year (6 issues)	$18.00	$24.00	$38.00	_____
2 years (12 issues)	$32.00	$44.00	$72.00	_____
Back Issues				
18 thru #43	$3.50 ea.		$5.00 ea.	_____
6 or more	$3.00 ea.		$4.50 ea.	_____
#44 and up	$4.50 ea.		$6.00 ea.	_____
6 or more	$4.00 ea.		$5.50 ea.	_____

Back Issues Ordered:

_____

Subscription Total _____
Back Issues Total _____
California state Residents add 7.25% Sales TAX _____
Total Enclosed _____

Name: _____
Address: _____
_____
_____
My Interests: _____
_____

Payment is accepted by check or money order. Checks must be in US funds, drawn on a US bank. Personal checks within the US are welcome.

TCJ The Computer Journal

P.O. Box 535, Lincoln, CA 95648-0535
Phone (916) 645-1670

Embedded Systems

Intermediate

D/A on the Cheap

Analog Signal Generation

By Al Mitchell

When calling AM Research, you will talk to "Mitch." That is Al Mitchell himself; designer, engineer, and Forth programmer. Mitch has designed and built several 80451 development systems. He uses FPC for the basis of his embedded and interactive development platform. A special Forth 80451 kernel resides on the chip and you debug and develop your product using code written on a PC that is then down-loaded and run under your control. It is pretty close to something I might have developed myself (given lots of time and money - but then I could also be president.)

Fortunately I don't need to develop it myself, as Mitch has done a very good job. Part of his package includes sample code and projects. I got him to give us the low down on how he produces DTMF analog signals (one of the sample projects included in his manual.) You could do this using $3 to $10 worth of D/A conversion, or you do it his way for 30 to 40 cents worth of easily available parts.

I listed this as an intermediate project due to the op amp design, but beginners will be able to figure out what is happening if you dust off a textbook on amplifiers (find the section on summing amps.) Actually this is a rather easy concept to get if you just hang in there and read on, so read on! BDK.

A recurring theme in our applications work has been to generate signals under the control of a microprocessor. While this is relatively easy using a DSP or other powerful processor, doing so with cost-effective components requires finesse and elegance rather than brute horsepower.

One such example is a remote, standalone data acquisition system which needs to download data on occasion. This application required only a single 8051-type processor for the task. Since it also had to dial the telephone, generating Dual Tone Multi-Frequency (DTMF) signals, the software required more than conventional techniques.

In this first of a series of articles we will discuss the theory of creating the precise analog counterpart to a digital data byte. The next article will describe the program and code requirements to drive the circuit in real time. Following articles will discuss alternative methods appropriate for differing applications.

Two methods of Digital-to-Analog (D/A) conversion can be constructed with relative ease. The first is commonly called a "Weighted Resistive Ladder" wherein each digital signal input (or ladder rung) has half the resistance of it's predecessor. The inputs are derived from the parallel port of the microprocessor used. A voltage (analog voltage) is produced which is equal to the additive voltage/current produced across all the resistors in the ladder.

Whether voltage/current is added into the circuit, depends on the state of the parallel line attached to a given resistor. The amount of additive signal is controlled by the voltage drop across the individual resistor. The minimum and maximum values of resistance are therefore related by a binary relationship, I. E.: 2^n. In this case we are constructing an eight bit A/D so the relationship of the largest resistive value is 2^8 or 256 times the value of the smallest resistor.

Both the maximum and minimum values are limited by components and construction method used. In our case, constructing a prototype wire wrapped PCB, the maximum reasonable value you can use is about one Meg ohm due to humidity, stray capacitance and availability of components. I prefer to use less than the maximum value for stability and reliability reasons. Lets check the minimum value which is 1/256 of Rmax or roughly 4K ohms. The first pass of our design is shown in figure 1.

Using the quasi-bidirectional port of the 8051 reveals that the output structure looks like a 30K pull-up resistor with an open-drain MOSFET pull-down (the internal components of the 8051.) This is a serious limitation since the 30K pull-up would be added to value of our resistor selection giving unacceptable performance. This can be overcome by using a CMOS driver, like the 74HC04 which will buffer the 8051 and provide very low output impedance of roughly 100 ohms. With proper drive we can now plug in the resistor values.

The output voltages can be easily determined by considering the input of the summing amplifier to be one-half the full scale voltage of Vcc or 2.5 volts. In describing the circuit operation we need to keep in mind that we will be generating a sine wave oscillating between Vcc and ground. This means that our reference is better described as 2.5 volts. The 2.5 volt reference is produced by using a low pass filter to average the sine wave

and to provide feed back to the op amp non-inverting input.

R1 then either provides a positive or negative current, relative to the summing node of the operational amplifier, of (Vcc - Vcc/2)/1K or 2.5 milliamperes. The op amp operation is to invert attempts to counterbalance this input current by producing an output which brings the inputs to an equal value (the results of feed back signal.) Therefore the output swings to a value of 2.5 Volts plus or minus (470*2.5ma or feedback resistor times current) 1.18 volts. Therefore the MSB drives the output to either 3.68 or 1.33 volts. Additional inputs to the network will combine to provide an output swing of roughly 0.15 to 4.85 volts.

Previously we determined that the maximum resistor size can be as high as one Meg ohm, circuit stability can be improved by using lower values so let us select 1.0K for the Most Significant Bit (MSB). It then follows that each successive resistor should be twice that value. Since most of us only have easy access to 5% resistors we must construct the resistors needed from easily available and standard sizes. Below is a listing of our selections:

Calculated Value	Rough Approximation	Finely Tuned
R1 = 1K	1K	910 ohm
R2 = 2K	2K	1.8K
R3 = 4K	3.9K	3.9K
R4 = 8K	7.5K + 510 ohms	7.5K
R5 = 16K	15K + 1K	15K
R6 = 32K	27K + 5.1K	27K
R7 = 64K	62K + 2K	62K
R8 = 128K	120K + 8.2K	120K

R9 = 470 ohms
R10 = 1Meg
C1 = 10 ufd

Note that the column entitled "Finely Tuned" uses a common trick of trimming standard resistances to the exact value needed using a needle file and DVM. Although this circuit is not critical in the need for specific resistor types, carbon composition are easiest to trim, they have a carbon granule core which is fairly soft and easy to grind away. Carbon and Metal Film resistors are vapor deposited upon a ceramic core and considerably more difficult to modify. All can be altered however by removing conductive material which increases the resistance.

Using a Digital Volt Meter we were able to get the exact resistance needed by a combination of selected series resistors. Should the selection not be available simply file the body of the resistor until the value desired is achieved, just make sure to seal the injury to prevent moisture contamination. If you go slightly too far the carbon varieties can be lowered a small amount with the judicious use of a hot soldering iron tip at the contact point. Note that resistors modified in this manner may change resistance slightly when soldered so it is a good practice to heat and stabilize them anyway.

The advantage of the weighted resistor network is that anyone with simple equipment can quickly construct an accurate D/A converter without rare component values.

A second economical method of D/A conversion uses the "R-2R" method. In this configuration only two resistances are needed regardless of the ultimate bit length required and therefore is most common in high resolution designs and on dedicated D/A integrated circuits. The wide disparity of resistance values, and temperature coefficients, are significantly decreased resulting in better accuracy.

The limitations are similar to the "Weighted Ladder" above as to drive requirements but the use of one value eases construction considerably. In place of the "2R" component two resistors can be used in series. Therefore three resistors of one value are needed per bit of input.

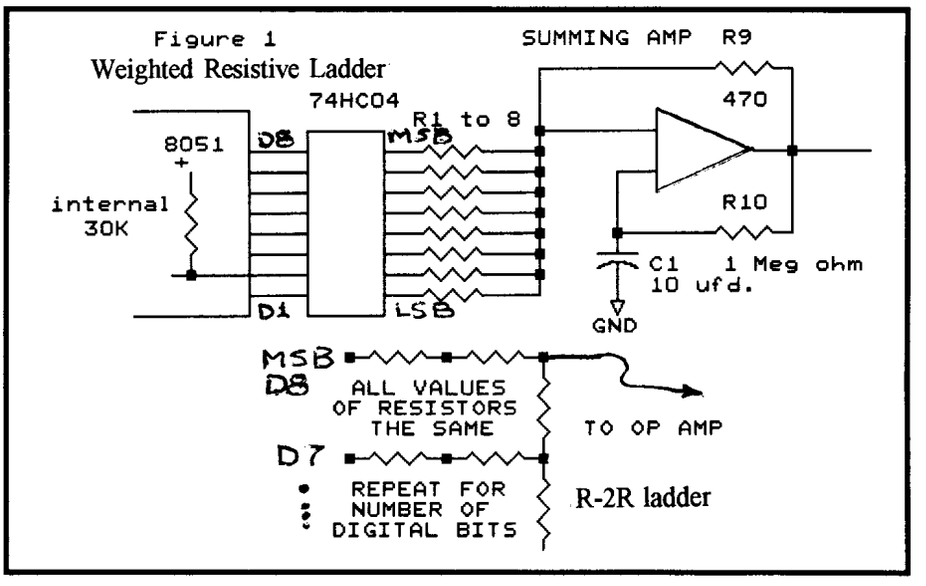
This is an excellent circuit when time, or bench stock, permits the use of standard precision parts. Note that the precision required is quite substantial with tolerance buildup although 1% components are usually sufficient for an 8-bit D/A.

The calculation of specific digital values is not difficult. Beginning with the summing amplifier reference equivalent to half scale, one can see that if the MSB is low (up output inverted) then the output of the summing amplifier is -(Vcc/2R)*R plus or minus the binary values of the remaining bits.

The advantage of the R-2R ladder D/A converter is that one single resistor value is used which can be constructed without special techniques.

Other more esoteric methods of signal generation will be covered in future columns.

Al Mitchell runs AM Research from 4600 Hidden Oaks Lane, Loomis, CA 95650 (call (916) 652-7472. He has been building embedded system for many years. When not building you can find him in his large garden, working away the hours.



Regular Feature

Editorial Comment

Peer-To-Peer LANs

Computer Corner

By Bill Kibler

Well our house move is completed but organization is far behind. I have a new home office and shop combination that will be pretty good when it gets all set up. That "when" however could be sometime long in the future as there are plenty of more important items in the queue to be done first.

For work, my direction has moved away from the 68000 and into LAN problems. We have received the latest version of LanMan 2.1 and are working with the problems it created. I have also been spending considerable time with Novell's NetLite product.

NetLite

Netware products have been mostly aimed at the major LAN user. The problem with that is the cost associated with even a two or three user network. Novell has no price break in essence for size of network. Complexity of Netware products is rather high as well. Special schools are necessary to service the LAN as their manuals do not explain all. Some changes have been made over the past few years but if you are a small user, generally it is more problems than it is worth.

Starting with the $25 network and now with several vendors making LAN based systems for the small user, Novell has entered the market. The $25 network is a serial based system for up to three users. I have it and it works just fine for small setups. For larger groups up to 25 users Netlite is now available. For more than 25 users, there are other systems. Our company did test three of the products and found Netlite the poorest of the group. At this stage I would recommend any of the other products as I will explain.

Netlite installs easily and does seem to work as advertised. If you happen to have any problems however, there is no way to get answers. Novell's own technical support group has not had a chance to learn about the product yet. Also the manuals are absolutely no help if it doesn't work. Our problem has to do with NETBIOS calls, and as I found out may be a problem in all Netware products.

We use NETBIOS operation to pass data between our mainframe and remote terminals. We have had some problems with vendors implementation of NETBIOS. NETBIOS was IBM's orignial method of interfacing to the LAN. It since has become a hardware and manufactures independent interface to the LAN network. However each vendor must implement their own version of the standard network interface. It is very similar in concept to DOS calls. NETBIOS has a stnadard interrupt you call with a pointer to a NCB, NETBIOS CONTROL BLOCK. The NCB contains pointers to where the data is, who it goes to, what operation is to be performed, and where to post the results of that operation.

As a programmer you can write code to this interface and should have the same results on any vendors setup. We have found that not to be completely true. Some of IBMs implementation stole keystrokes or turned off our interrupts. 3COMs old version of NETBIOS bounced when first started. 3COM has solved the problems in their latest release. Netwares NETBIOS appears to work fine on Token ring but fails on Ethernet systems. If you are running a versions of these products you might try COMPUSERVE. We have downloaded new versions of 3COMs and Novell NETBIOS programs from there.

Speaking of downloading, I have found out that you could make a complete LAN system by using FREE upgrades from compuserve. In working with NetLite I discovered that the device drivers and NETBIOS driver are avaialable to be downloaded. They are intended as upgrades for their current users. They will work as a NETBIOS system as they are. This means no security, or fancy DOS interfaces. However combined with other FREE NETBIOS programs it is possible to have a system that can pass files between stations, run programs remotely, send message to other users, and even create multi-user type systems.

I have tried several of the free NETBIOS programs with varying results. Some worked very well, others only locked up and died. There is of course no support for these systems, but the cost is very reasonable. I plan on trying to pickup some older Ethernet cards and try my hand at multiple processor operations using NETBIOS calls. Since I program using NETBIOS operations that should be a easy task. For most, the programs on the BBSs or from books will have to do. There appears to be only two books available. "C programmer's Guide to NETBIOS", by David Schwaderer, and "Network programming in C" by Barry Nance.

I have bought Nance's book and it comes with programs on disk. The programs work and his book makes an excellant reference manual. Our comapany product was built using help from Schwaderer,

and disks are available by mail. The references in Schwaderer are very good on regular NETBIOS operations, but Nance also covers Novell's IPX/SPX interface. Examples in either book do appear to work and provide a sound basis for making your own NETBIOS to NETBIOS system.

The Netlite structure is based on TSR device drivers. These TSRs programs provide communications with the LAN interface card and NETBIOS. For Netlite a Client and Server TSR provide interfaces into DOS creating the peer-to-peer network. Most big networks work off the idea of a single server computer that controls who and what users can do. All common files are maintained on the server and other users normally would not have access to other users data without it first being put on the server hard drive. The server as such becomes a very small mainframe system. This means one system is dedicated for LAN control and data storage only.

In small organizations, the use of the LAN is for message passing, sharing of printers, and limited data or file sharing. Many small organizations would be happy to eliminate hand carrying of floppy disks from system to system. The peer-to-peer LAN systems are designed for this market. No severs are needed or all users can be servers. The design concept is that all users have the same options and can setup their station as the situation warrants.

Other magazines have reviewed these small LANs and most like Novell's approach to explaining the how and why of peer-to-peer systems. Cost is basically $99 per user including hardware(LAN interface card). Novell was able to produce their product quickly and cheaply by only writing two new programs. Their Client and Server programs appear to be cut down versions of other parts of Netware systems. This means that all the device drviers are the exact same ones as used on the bigger Netware system. In fact they use ODI drivers completely. ODI is a standard device driver interface that you could create yourself if needed. I have been thinking about doing just that for a serial based ring, but that is another subject.

The bottom line is we have found a very special problem in Netlite and Netware in general. It appears that given data packets back-to-back the NETBIOS interface will ABEND or die. It ABENDs so bad that in most cases you must power down the system to bring it back on line. Doing so of course prevents you from checking your data in anyway to find out what happened. I modified our device drivers with EFORTH and was able to do some crude debugging even after an ABEND. The only way out of the device driver after running EFORTH and getting my data was the power switch.

If you are starting to get the idea that working with LANs is complex, expensive, and problemmatic, you are getting the correct idea. Novel called about the problem and wonder if we could send over some of our stuff so they could see for themself what the problem is. My comment was a question, it needs a mainframe, special interface software, several stations, or about a quarter million dollars in hardware and software systems to make it work. As I have said before, working on LANs and most new systems is no longer a little garage type operation. These systems cost money, lots of money, and take lots of time and resources to make work.

68HC16

To move on to another good topic is Motorolas newest trainer board. I bought one of the DSP (digital signal processing) trainer kits. For $168 it is a great buy. So when you get tired of fighting the big boys, try these little boxes. The software and hardware is excellant. The HC16toolkit is an excellant trainer. I have already started on the books, and believe me there were plenty of books included.

You get two boards in this set, the 68HC16 evaluation board and an stereo signal interface board with display. The project looks like fun and I will report more about it later. If you did not get one, try and borrow one from a friend. When it comes to supporting their users, hardware users in this case, Motorolla still is number one in my book.

ENOUGH

I said enough this time and had enough to do with LANs for the day. Next time I will explain more about the HC16 trainer and its features. I hope to have finished the project with it and may be able to comment on DSP system better after the project. Keep hacking.

In Issue #57

Everything you always wanted to know about X-Modem protocols.

8-Bit D/A on the Cheap for DTMF tones.

ZED-Fest in Europe Report.

The inside report on LAN protocol formats.

A Beginners Guide to using FPC for embedded development.

IN THE WORKS:
S-100 update.

8031 for the beginner.

Fundamentals of Stepper motors.

6805, better than 8031?

Communicating between 8031 processors.

68332 an ideal engine for embedded systems?

Budget project development - how to use FREE CAD software.

Interface expansion for T-800.

Serial based Data Acquisition.

Z-80 still the best embedded controller?

TCJ The Computer Journal Market Place

Discover
The Z-Letter

The Z-Letter is the only monthly publication for CP/M and Z-System. Eagle computer and SpellBinder support. Licensed CP/M distributor.

Subscriptions: $15 US, $18 Canada and Mexico, $45 Overseas

Write or call for free sample.

The Z-Letter
Lambda Software Publishing
720 South Second Street
San Jose CA 95112-5820
(408) 293-5176

Advent Kaypro Upgrades

TurboROM. Allows flexible configuration of your entire system, read/write additional formats and more. $35

Hard drive conversion kit. Includes interface, controller, TurboROM, software and manual—Everything needed to install a hard drive except the cable and drive! $175 without clock, $200 with clock.

Personality Decoder Board. Run more than two drives, use quad density drives when used with TurboROM. $25

Limited Stock — Subject to prior sale

Call 916-483-0312 eves weekends or write Chuck Stafford, 4000 Norris Avenue, Sacramento CA 95821

TCJ Market Place
Advertising for Small Business

Looking for a way to get your message across? Advertise in the Market Place!

First Insertion:	$50
Reinsertions:	$35

Rates include typesetting. Payment must accompany order. Visa, MasterCard, Discover, Diner's Club, Carte Blanche, JCB, EuroCard accepted. Checks, money orders must be in US funds drawn on a US bank. Resetting of ad constitutes a new advertisement at first insertion rate. Inquire for rates for larger ads if required. Deadline is eight weeks prior to publication date. Mail to:

The Computer Journal
Market Place

CP/M SOFTWARE

100 page Public Domain Catalog, $8.50 plus $1.50 shipping and handling. New Digital Research CP/M 2.2 manual, $19.95 plus $3.00 shipping and handling. Also, MS/PC-DOS Software. Disk Copying, including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog $1.00

Elliam Associates
Box 2664
Atascadero, CA 93423
805-466-8440

Kenmore
ZTime-1
Real Time Clocks

Assembled and Tested with
90 Day Warranty
Includes Software

$79.95

Send check or money order to
Chris McEwen
PO Box 12
South Plainfield, NJ 07080
(allow 4-6 weeks for delivery)

Z-System Software Update Service

Provides Z-System public domain software by mail.

Regular Subscription Service
Z3COM Package of over 1.5 MB of COM files
Z3HELP Package with over 1.3 MB of online documentation
Z-SUS Programmers Pack, 8 disks full
Z-SUS Word Processing Toolkit
And More!

For catalog on disk, send $2.00 ($4.00 outside North America)
and your computer format to:

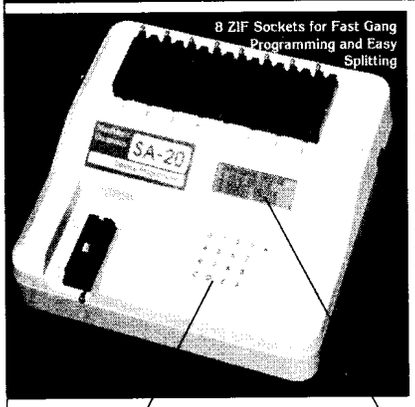
Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469

Major Upgrade Announcement
Z3Help and Z3COM Packages
Write for Details

Major Upgrade Announcement
Z3Help and Z3COM Packages
Write for Details

EPROM PROGRAMMERS

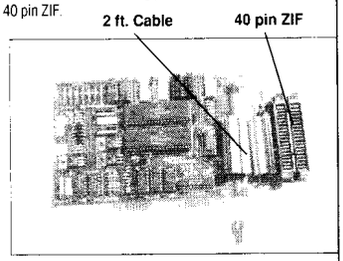
Stand-Alone Gang Programmer $750.00


8 ZIF Sockets for Fast Gang Programming and Easy Splitting
SA-20
20 Key Tactile Keypad (not membrane)
20 x 4 Line LCD Display

- Completely stand-alone or PC driven
- Programs E(E)PROMs
- ***1 Megabit of DRAM***
- ***User upgradable to 32 Megabit***
- ***.3/.6" ZIF socket, RS-232, Parallel In and Out***
- 32K internal Flash EEPROM for easy firmware upgrades
- ***Quick Pulse Algorithm (27256 in 5 sec, 1 Megabit in 17 sec.)***
- 2 year warranty
- Made in U.S.A.
- Technical support by phone
- Complete manual and schematic
- ***Single Socket Programmer also available. $550.00***
- Split and Shuffle 16 & 32 bit
- 100 User Definable Macros, 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex, and Motorola S

Internal Programmer for PC $139.95

New Intelligent Averaging Algorithm. Programs 64A in 10 sec., 256 in 1 min., 1 Meg (27010, 011) in 2 min. 45 sec., 2 Meg (27C2001) in 5 min. Internal card with external 40 pin ZIF.



- Reads, verifies, and programs 2716, 32, 32A, 64, 64A, 128, 128A, 256, 512, 513, 010, 011, 301, 27C2001, MCM 68764, 2532
- ***Automatically sets programming voltage***
- Load and save buffer to disk
- Binary, Intel Hex, and Motorola S formats
- ***Upgradable to 32 Meg EPROMs***
- ***No personality modules required***
- 1 year warranty • 10 day money back guarantee
- Adapters available for 8748, 49, 51, 751, 52, 55, TMS 7742, 27210, 57C1024, and memory cards
- Made in U.S.A.

NEEDHAM'S ELECTRONICS
4539 Orange Grove Ave. • Sacramento, CA 95841
Mon. - Fri. 8am - 5pm PST

Call for more information
(916) 924-8037
FAX (916) 972-9960

C.O.D.

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low as $100.00
Developer Packages
as low as $200.00(a $50.00 Savings)

A New Project
Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files.

Get It To Market--FAST
Don't wait until the hardware is finished to debug your software. Our Simulators can test your program logic before the hardware is built.

No Source!
A minor glitch has shown up in the firmware, and you can't find the original source program. Our line of disassemblers can help you re-create the original assembly language source.

Set To Go
Buy our developer package and the next time your boss says "Get to work.", you'll be ready for anything.

Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

BROAD RANGE OF SUPPORT
- Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	Intel 80C196

- All products require an IBM PC or compatible.

So What Are You Waiting For? Call us:

PseudoCorp
Professional Development Products Group
716 Thimble Shoals Blvd, Suite E
Newport News, VA 23606
(804) 873-1947 FAX: (804)873-2154

William P Woodall • Software Specialist

Custom Software Solutions for Industry:
- Industrial Controls
- Operating Systems
- Image Processing
- Hardware Interfacing
- Proprietary Languages
- Component Lists

Custom Software Solutions for Business:
- Order Entry
- Warehouse Automation
- Inventory Control
- Wide Area Networks
- Point-of-Sale
- Accounting Systems
- Local Area Networks
- Telecommunications

Publishing Services:
- Desktop Systems
- Books
- CBT
- Format Conversions
- Directories
- Interactive Video

33 North Doughty Ave, Somerville, NJ 08876 • (908) 526-5980

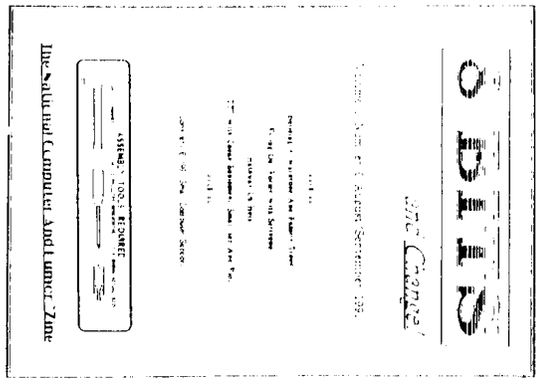
$15 / year $18 / year Canada

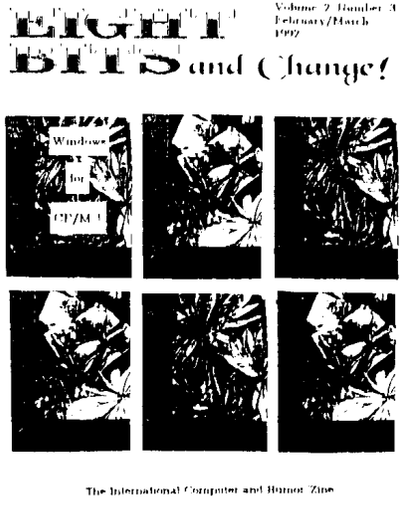
$21 / year overseas

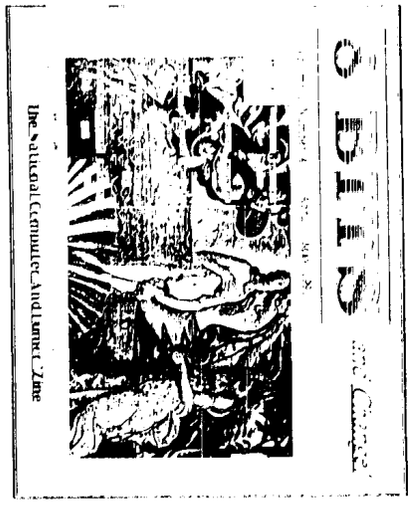
Small Computer Support

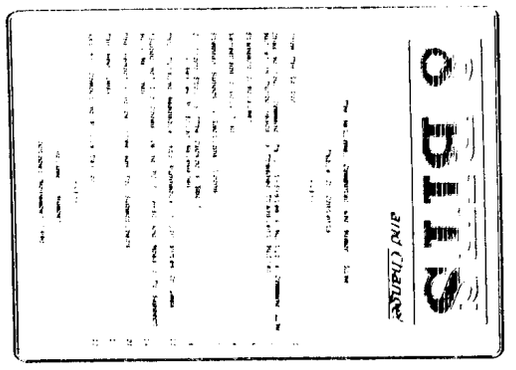
24 East Cedar Street

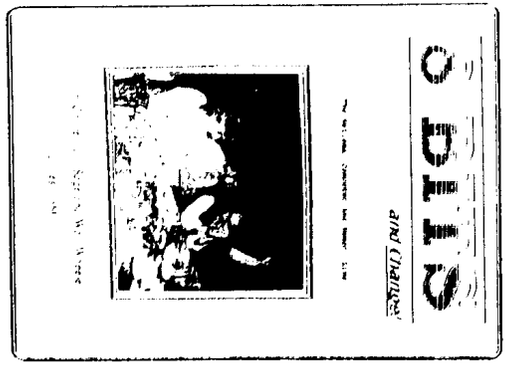
Newington, CT 06111





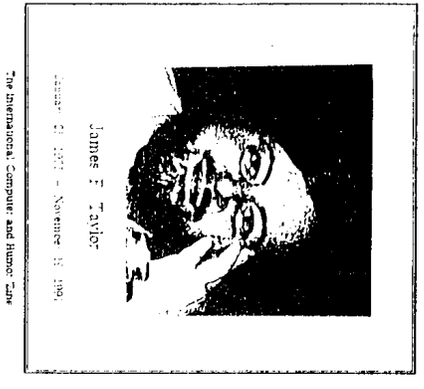


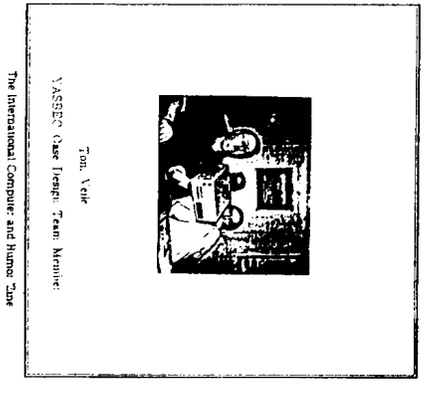


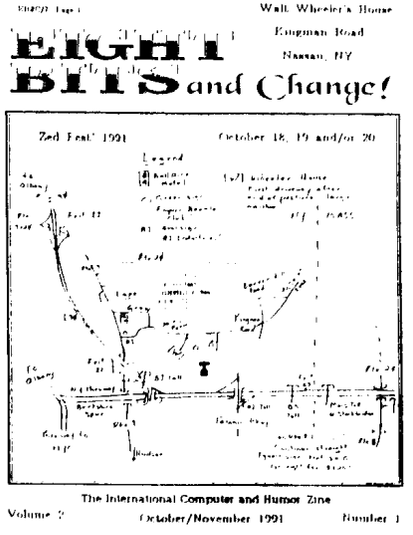


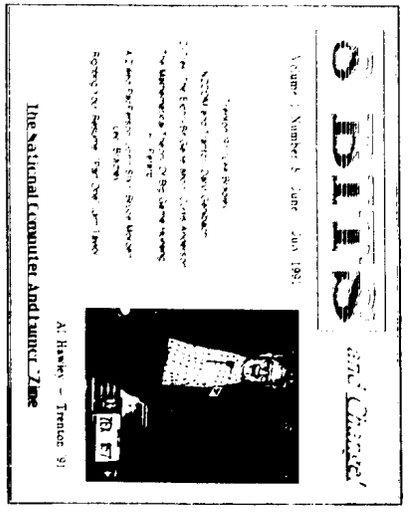
Have you been missing this ????

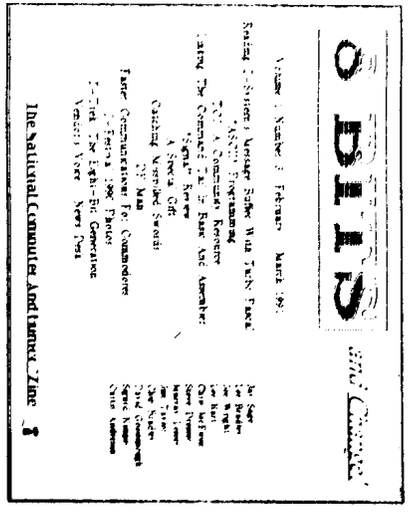
All 10 back issues: $40











203 666–3139 (voice) 203 665–1100 (modem)

Back issues: $5 apiece

The Forth Interest Group

continues to be

the best Forth resource

Forth publications and book library — The Forth Interest Group (FIG) carries the largest selection found anywhere—reference books, public-domain Forths, tutorials and tools.

Forth Dimensions — Members' bi-monthly magazine is devoted exclusively to Forth. It offers techniques, advanced language proposals, source code, Forth product and business news, and extensive resource listings.

FIG chapters — An opportunity for local, face-to-face meetings with other Forth enthusiasts and professionals. Technical seminars, vendor presentations, tutorials and group hardware/software projects have been some chapters' activities.

Fidonet™ RoundTable — A central focus for on-line technical discussion, and access to Forth users, vendors, developers. Includes a library of more than 700 downloadable files.

Annual FORML conference — Participate in technical sessions and mingle with leading Forth experts in an informal setting at the Asilomar Conference Center, located on the California coastline in Pacific Grove.

FIG is a non-profit, membership organization of over 1500 members in 20 countries. Membership includes a subscription to *Forth Dimensions*, discounts on Forth literature, and more. Join us today!

Yes! I want to join the world's foremost Forth programmers and receive the bi-monthly magazine Forth Dimensions.
Enclosed payment for a one-year membership in the Forth Interest Group:
$30 U.S.A./Canada/Mexico • $46 Canada air mail • $52 overseas air mail
Student rate $18 (full-time students, copy of student I.D. required)

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____

Daytime telephone _____

☐ Visa ☐ MasterCard _____

exp. date _____

Please enclose this form and your check or money order (or Visa/MasterCard numbers) in an envelope to the:

Forth Interest Group
P.O. Box 8231 • San Jose, California 95155 • USA
(Call 408-277-0668 or fax 408-286-8988)

Forth fits...

When memory, time, and budget are tight, or when your first-choice solution cries out for an extensible language, think of Forth. It's perfect for embedded systems, and is *legendary* for fitting in small places...even places in orbit on the space shuttle. Today, Forth is at home under most any operating system, and implementors offer both lean and rich environments. Join the Forth Interest Group now and learn how well Forth fits in your programming toolbox.